EXHIBIT D4

RTCA, Inc.
1150 18th Street, NW, Suite 910
Washington, DC 20036

Model-Based Development and Verification Supplement to DO-178C and DO-278A

RTCA DO-331
December 13, 2011

Prepared by: SC-205

Copies of this document may be obtained from

RTCA, Inc.

Telephone: 202-833-9339
Facsimile: 202-833-9434
Internet: www.rtca.org

Please visit the RTCA Online Store for document pricing and ordering information.

FOREWORD

This document was prepared by RTCA Special Committee 205 (SC-205) and EUROCAE Working Group 71 (WG-71) and approved by the RTCA Program Management Committee (PMC) on December 13, 2011.

RTCA, Incorporated is a not-for-profit corporation formed to advance the art and science of aviation and aviation electronic systems for the benefit of the public. The organization functions as a Federal Advisory Committee and develops consensus-based recommendations on contemporary aviation issues. RTCA's objectives include but are not limited to:

- coalescing aviation system user and provider technical requirements in a manner that helps government and industry meet their mutual objectives and responsibilities;
- analyzing and recommending solutions to the system technical issues that aviation faces as it continues to pursue increased safety, system capacity, and efficiency;
- developing consensus on the application of pertinent technology to fulfill user and provider requirements, including development of minimum operational performance standards for electronic systems and equipment that support aviation; and
- assisting in developing the appropriate technical material upon which positions for the International Civil Aviation Organization and the International Telecommunication Union and other appropriate international organizations can be based.

The organization's recommendations are often used as the basis for government and private sector decisions as well as the foundation for many Federal Aviation Administration Technical Standard Orders.

Since the RTCA is not an official agency of the United States Government, its recommendations may not be regarded as statements of official government policy unless so enunciated by the U.S. government organization or agency having statutory jurisdiction over any matters to which the recommendations relate.

This Page Intentionally Left Blank

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

This Page Intentionally Left Blank

MB.1.0 INTRODUCTION

A model is an abstract representation of a set of software aspects of a system that is used to support the software development process or the software verification process. This supplement addresses model(s) that have the following characteristics:

a. The model is completely described using an explicitly identified modeling notation. The modeling notation may be graphical and/or textual.

b. The model contains software requirements and/or software architecture definition.

c. The model is of a form and type that is used for direct analysis or behavioral evaluation as supported by the software development process or the software verification process.

The following are not considered as models within this supplement:

- Figures without syntax/semantics (these may be illustrative).
- Equations related to natural language sentences.

The use of models may bring the benefits and capabilities of:

- Providing unambiguous expression of requirements and architecture.
- Supporting the use of automated code generation.
- Supporting the use of automated test generation.
- Supporting the use of analysis tools for verification of requirements and architecture.
- Supporting the use of simulation for partial verification of requirements, architecture, and/or Executable Object Code.

Since the publication of DO-178B, advances and experience have been gained in model-based development and verification, their application, and supporting tools. As the use of this technology for critical software applications in avionics has increased, there are a number of issues that need to be considered to ensure the safety and integrity goals are met. Clarifying each of these issues will ease the application of model-based development and verification; therefore, this supplement provides guidance for applicants and certification or approval authorities to facilitate the use of this technology.

MB.1.1 Purpose

This supplement contains modifications and additions to DO-178C objectives, activities, explanatory text, and software life cycle data that should be addressed when model-based development and verification are used as part of the software life cycle. This includes the artifacts that would be expressed using models and the verification evidence that could be derived from them. Therefore, this supplement also applies to the models developed in the system process that define software requirements or software architecture.

MB.1.2 Scope

This supplement discusses the use of model-based development and verification in the software life cycle for software that is produced in accordance with DO-178C. If the applicant is planning to use model-based development and verification, then the applicant

should comply with this supplement. This supplement does not provide guidance for models developed and/or used only for the software verification process.

MB.1.3 Relationship to Other Documents

This document supplements the guidance given in DO-178C. This document may be used in conjunction with other DO-178C supplements.

MB.1.4 How to Use This Document

This document should be used with and in the same way as DO-178C; however, the following should be noted:

a. Section MB.1 contains explanatory text to aid the reader in understanding model-based development and verification, and therefore is not to be taken as guidance.

b. Annex MB.A of this supplement describes how the DO-178C objectives are revised in line with this model-based development and verification guidance.

c. Guidance from DO-178C should be used for all aspects of the software life cycle, apart from where model-based development and verification are used. All guidance specific to model-based development and verification is contained within this supplement.

d. This supplement can be applied to DO-278A in the same manner as DO-178C. Applicants complying with DO-278A should be aware that the text within this supplement is based on DO-178C; thus the DO-278A applicant needs to ensure that the proper DO-278A guidance is being addressed. Annex MB.C of this document describes how the DO-278A, Annex A objectives are revised in line with model-based development and verification guidance in this supplement.

e. In addition to the guidance contained in this supplement, this supplement also provides clarification on the guidance provided in the form of answers to Frequently Asked Questions (FAQs) and Discussion Papers (DPs) (see Appendix MB.B). Neither the FAQs nor the DPs contain new or additional guidance.

f. Major sections are numbered as MB.X.0 throughout this supplement. It should be noted that references to an entire section are identified as “section MB.X”; whereas, references to the content between section headers MB.X.0 and MB.X.1 are referred to as “section MB.X.0”.

MB.1.5 Document Overview

This supplement is intended to be used with DO-178C. Sections MB.2 through MB.12 and Annex MB.A provide additional and modified text to the referenced sections of DO-178C. The document strives to minimize redundancy between DO-178C and the document itself. Each affected section of DO-178C is included in this document. Where the guidance from DO-178C applies unchanged, this is stated in this document. Text taken directly from DO-178C is shown in italics. For those sections from DO-178C included in this supplement that are unchanged, except for the references, the DO-178C text is included with the updated MB reference.

In Annex MB.A, only those tables with changes are included. These tables include new objectives, activities, and changes to referenced text for existing objectives.

MB.1.6 Characteristics of Model-Based Development and Verification

As part of the software development and verification processes, models may represent software requirements and/or architecture, partially or completely.

Models are characterized by the modeling technique that is used and by the type of software life cycle data they will represent. A modeling technique is a combination of a modeling language and a particular manner of using this modeling language. The ability of a model to correctly and completely express a certain type of software life cycle data depends on the modeling technique used. Modeling techniques used should be suitable to the type and to the level of abstraction of the information expressed by the model. Software Model Standards are the means to describe a modeling technique and support the demonstration of how this technique is fit for its purpose.

MB.1.6.1 Requirements from Which the Model is Developed

For any type of model within the scope of this supplement, there is a need to identify the requirements from which the model is developed. Requirements from which the model is developed should be external to the model. Those requirements should provide details and constraints to enable model development and verification activities.

MB.1.6.2 Specification Models and Design Models

This supplement introduces two types of models: Specification Models and Design Models.

A Specification Model represents high-level requirements that provide an abstract representation of functional, performance, interface, or safety characteristics of software components. The Specification Model should express these characteristics unambiguously to support an understanding of the software functionality. It should only contain detail that contributes to this understanding and does not prescribe a specific software implementation or architecture except for exceptional cases of justified design constraints. Specification models do not define software design details such as internal data structures, internal data flow or internal control flow. Therefore, a Specification Model may express high-level requirements but neither low-level requirements nor software architecture.

A Design Model prescribes software component internal data structures, data flow, and/or control flow. A Design Model includes low-level requirements and/or architecture. In particular, when a model expresses software design data, regardless of other content, it should be classified as a Design Model. This includes models used to produce code.

A model cannot be classified as both a Specification Model and a Design Model.

For the purpose of this supplement, the term “high-level requirement” refers to either of the following:

- Any requirement contained in a Specification Model.
- Any requirement from which a Design Model is developed.

For the purpose of this supplement, the term “low-level requirement” refers to:

- Any requirement contained in a Design Model.

MB.1.6.3 Examples of Model Usage

Table MB.1-1 reflects some of the current industry practices used for the development of software requirements and/or architecture.

Table MB.1-1 Model Usage Examples

Process that generates the life-cycle data	MB Example 1	MB Example 2	MB Example 3	MB Example 4 (See Note 1)	MB Example 5 (See Note 1)
System Requirement and System Design Processes	Requirements allocated to software	Requirements from which the Model is developed	Requirements from which the Model is developed	Requirements from which the Model is developed	Requirements from which the Model is developed
					Design Model
Software Requirement and Software Design Processes	Requirements from which the Model is developed	Specification Model (See Note 2)	Specification Model	Design Model	
	Design Model	Design Model	Textual description (See Note 3)		
Software Coding Process	Source Code	Source Code	Source Code	Source Code	Source Code

Note 1: In MB Example 4 and MB Example 5 it is difficult to separate system and software life cycle data. Regardless of where the model originated, the supplement applies and the objectives of this supplement should be satisfied. Therefore, in these examples the guidance related to low-level requirements should be applied to the Design Model and the guidance related to high-level requirements should be applied to the requirements from which the model is developed. See the explanatory text below Table MB.A-3 of Annex MB.A related to the applicability of objectives in this case.

Note 2: In MB Example 2, the Design Model is developed from the requirements contained in the Specification Model.

Note 3: In MB Example 3, the textual description refers to low-level requirements (and possibly software architecture); DO-178C guidance is applicable to these.

MB.2.0 SYSTEM ASPECTS RELATING TO SOFTWARE DEVELOPMENT

This section discusses those aspects of the system life cycle processes necessary to understand the software life cycle processes. System life cycle processes can be found in other industry documents (for example, SAE ARP4754A).

Discussed in this section are:

- *System requirements allocation to software (see* DO-178C section *2.1).*
- *The information flow between the system and software life cycle processes and between the software and hardware life cycle processes (*see MB.2.2*).*
- *The system safety assessment process, failure conditions, software level definitions, and software level determination (see* DO-178C section *2.3).*
- *Architectural considerations (see* DO-178C section *2.4).*
- *Software considerations in system life cycle processes (see* MB.2.5*).*
- *System considerations in software life cycle processes (see* DO-178C section *2.6).*

The term "system" in the context of this document refers to the airborne system and equipment only, not to the wider definition of a system that might include operators, operational procedures, etc.

MB.2.1 System Requirements Allocation to Software

Section 2.1 of DO-178C is unchanged.

MB.2.2 Information Flow Between System and Software Life Cycle Processes

Figure MB.2-1 *is an overview of the information flow between system life cycle processes and the software life cycle processes. This information flow includes the system safety aspects. Due to interdependence of the system safety assessment process and the system design process, the flow of information described in these sections is iterative.*

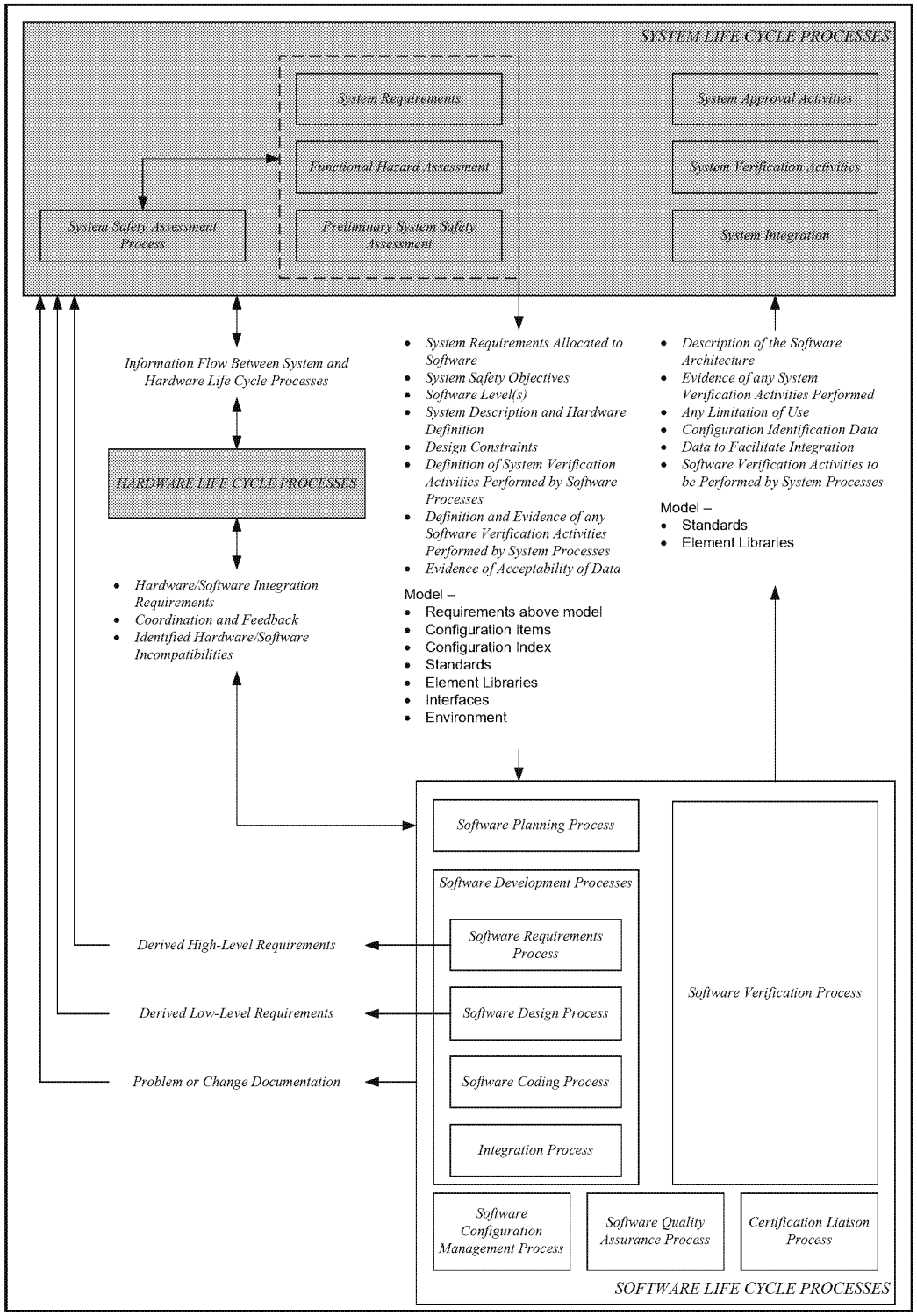

Figure MB.2-1 Information Flow Between System And Software Life Cycle Processes

MB.2.2.1 Information Flow from System Processes to Software Processes

The following data is passed to the software life cycle processes by the system processes either as part of the requirements allocation or during the development life cycle:

a. System requirements allocated to software.

b. System safety objectives.

c. Software level for software components and a description of associated failure condition(s), if applicable.

d. System description and hardware definition.

e. Design constraints, including external interfaces, partitioning requirements etc.

f. Details of any system activities proposed to be performed as part of the software life cycle. Note that system requirement validation is not usually part of the software life cycle processes. The system life cycle processes are responsible for assuring any system activities proposed to be performed as part of the software life cycle.

g. Evidence of the acceptability, or otherwise, of any data provided by the software processes to the system processes on which any activity has been conducted by the system processes. Examples of such activity are the system processes' evaluations of:

1. Derived requirements provided by the software processes to determine if there is any impact on the system safety assessment and system requirements.

2. Issues raised by the software processes with respect to the clarification or correction of system requirements allocated to software.

h. Evidence of software verification activities performed by the system life cycle processes, if any.

If system processes provide a Specification Model or Design Model to the software processes, then the following data should be provided as inputs to the software processes:

i. Requirements from which the model was developed.

j. Model configuration items (files or data that represent the model).

k. Model standards describing the modeling techniques (see MB.11.23).

l. Model element libraries.

m. Model and system interfaces description.

n. Configuration index of the model configuration items.

o. Modeling development environment and user's manuals.

p. Any data from verification and/or validation activities performed at system level that may be used to satisfy verification objectives as defined in section MB.6.

Any evidence provided by the system processes (see MB.2.2.1.f *and* MB.2.2.1.g*) should be considered by software processes to be Software Verification Results (see* MB.11.14*).*

MB.2.2.2 Information Flow from Software Processes to System Processes

The software life cycle processes analyze the system requirements allocated to software as part of the software requirements process. If such an analysis identifies any system requirements as inadequate or incorrect, the software life cycle processes should capture the issues and refer them to the system processes for resolution. Furthermore, as the software design and implementation evolves, details are added and modifications made that may affect system safety assessment and system requirements.

To aid the evaluation of the evolving design and changes to the design, the software life cycle processes should make data available to the system processes including the system safety assessment process. This data will facilitate analyses and evaluations to establish the effect on the system safety assessment and system requirements. It may be advantageous for such analyses and evaluations to be performed jointly by the systems and software processes. Such data includes:

a. Details of derived requirements created during the software life cycle processes.

b. A description of the software architecture, including software partitioning.

c. Evidence of system activities performed by the software life cycle processes, if any.

d. Problem or change documentation, including problems identified in the system requirements allocated to software and identified incompatibilities between the hardware and the software.

e. Any limitations of use.

f. Configuration identification and any configuration status constraints.

g. Performance, timing, and accuracy characteristics.

h. Data to facilitate integration of the software into the system.

i. Details of software verification activities proposed to be performed during system verification, if any.

When the system processes provide a Specification Model or a Design Model to the software processes, modifications to the following data should be coordinated between the system development and software development processes. This data should be under the appropriate configuration control category for the software level:

j. Model standards describing the modeling techniques (see MB.11.23).

k. Model element libraries.

MB.2.2.3. Information Flow between Software Processes and Hardware Processes

Section 2.2.3 of DO-178C is unchanged.

MB.2.3 System Safety Assessment Process and Software Level

Section 2.3 of DO-178C is unchanged.

MB.2.4 Architectural Considerations

Section 2.4 of DO-178C is unchanged.

MB.2.5 Software Considerations in System Life Cycle Processes

Section 2.5 of DO-178C is unchanged.

MB.2.5.1 Parameter Data Items

Section 2.5.1 of DO-178C is unchanged.

MB.2.5.2 User-Modifiable Software

A user-modifiable component is that part of the software that may be changed by the user within the modification constraints without certification authority review, if the system requirements provide for user modification. A non-modifiable component is that which is not intended to be changed by the user. The potential effects of user modification are determined by the system safety assessment process and used to develop the software requirements, and then, the software verification process activities. Designing for user-modifiable software is discussed further in section MB.5.2.3. *A change that affects the non-modifiable software, its protection, or the modifiable software boundaries is a software modification and is discussed in section* MB.12.1.1.

Guidance for user-modifiable software includes:

a. The user-modifiable software should not adversely affect safety, operational capabilities, flight crew workload, any non-modifiable software components, or any software protection mechanism used. Unless this can be established, the software may not be classified as user-modifiable. The safety impact of displaying information based on user-modifiable software should also be considered.

b. When the system requirements provide for user modification, then users may modify software within the modification constraints without certification authority review.

c. The system requirements should specify the mechanisms that prevent the user modification from affecting system safety whether or not they are correctly implemented. The software that provides the protection for user modification should be at the same software level as the function it is protecting from errors in the modifiable component.

d. If the system requirements do not include provision for user modification, the software should not be modified by the user unless compliance with this document is demonstrated for the modification.

e. At the time of the user modification, the user should take responsibility for all aspects of the user-modifiable software, for example, software configuration management, software quality assurance, and software verification.

f. The applicant should provide the necessary information to enable the user to manage the software in such a way that the safety of the aircraft is not compromised.

MB.2.5.3 Commercial-Off-The-Shelf Software

COTS software included in airborne systems or equipment should satisfy the objectives of this document.

If deficiencies exist in the software life cycle data of COTS software, the data should be augmented to satisfy the objectives of this document. The guidance in section MB.12.1.4,

Upgrading a Development Baseline, and section MB.12.3.4*, Product Service History, may be relevant in this instance.*

MB.2.5.4 Option-Selectable Software

Some airborne systems and equipment may include optional functions that may be selected by software-programmed options rather than by hardware connector pins. The option-selectable software functions are used to select a particular configuration within the target computer. See MB.4.2.h, DO-178C sections *5.2.4,* and DO-178C section *6.4.4.3.d.2 for guidance on deactivated code.*

When software programmed options are included, a means should be provided to ensure that inadvertent selections involving non-approved configurations for the target computer within the installation environment cannot be made.

MB.2.5.5 Field-Loadable Software

Section 2.5.5 of DO-178C is unchanged.

MB.2.5.6 Software Considerations in System Verification

Section 2.5.6 of DO-178C is unchanged.

MB.2.6 System Considerations in Software Life Cycle Processes

Section 2.6 of DO-178C is unchanged.

MB.3.0 SOFTWARE LIFE CYCLE

Section 3.0 of DO-178C is unchanged.

MB.3.1 Software Life Cycle Processes

The software life cycle processes are:

a. *The software planning process that defines and coordinates the activities of the software development and integral processes for a project.* Section MB.4 *describes the software planning process.*

b. *The software development processes that produce the software product. These processes are the software requirements process, the software design process, the software coding process, and the integration process.* Section MB.5 *describes the software development processes.*

c. *The integral processes that ensure the correctness and control of, and confidence in the software life cycle processes and their outputs. The integral processes are the software verification process, the software configuration management process, the software quality assurance process, and the certification liaison process. It is important to understand that the integral processes are performed concurrently with the software planning and development processes throughout the software life cycle.* Sections MB.6 through MB. 9 *describe the integral processes.*

MB.3.2 Software Life Cycle Definition

Section 3.2 of DO-178C is unchanged.

MB.3.3 Transition Criteria Between Processes

Section 3.3 of DO-178C is unchanged.

This Page Intentionally Left Blank

MB.4.0 SOFTWARE PLANNING PROCESS

This section discusses the objectives and activities of the software planning process. This process produces the software plans and standards that direct the software development processes and the integral processes. Table MB.A-1 of Annex MB.A *is a summary of the objectives and outputs of the software planning process by software level.*

The applicant should ensure that the intended use of this supplement is acceptable to the appropriate certification authority. Several supplements may be applied to an applicant's software life cycle. The applicant should plan an approach that will comply with the objectives of DO-178C and those applicable objectives from the supplements. As such, all relevant supplements and the supplements' associated objectives should be considered. If the applicant's approach identifies a conflict between the objectives, then the applicant should propose a solution to the conflict in the Plan for Software Aspects of Certification (PSAC). The applicant may use a single set of software plans or multiple software plans addressing the different supplements. Using a single set of software plans, containing sections addressing each supplement, will provide an integrated approach with a clear and comprehensive plan.

As supplements only extend the guidance from DO-178C for a specific technology, the applicant should first consider all of the DO-178C objectives, and then the supplement's objectives, and lastly any other additional considerations. One approach is to consolidate all objectives of DO-178C and the applicable supplements, and for each objective provide a statement of how compliance will be achieved, along with identifying any applicable life cycle data items. Because the objective numbering scheme is unique between the Annex A Tables of DO-178C and the Annex A Tables of the supplements, the identification of the objectives and their document sources will be clear and unambiguous.

Specifically related to model-based development, it may be difficult to separate system and software life cycle data. The Specification Model or the Design Model may originate from the system requirements and design process. Regardless of where these models originate, the guidance of this supplement applies and the objectives of this supplement should be satisfied.

MB.4.1 Software Planning Process Objectives

The purpose of the software planning process is to define the means of producing software that will satisfy its requirements and provide the level of confidence that is consistent with the software level. The objectives of the software planning process are:

a. *The activities of the software development processes and integral processes of the software life cycle that will address the system requirements and software level(s) are defined (see* MB.4.2*).*

b. *The software life cycle(s), including the inter-relationships between the processes, their sequencing, feedback mechanisms, and transition criteria are determined (see* MB.3*).*

c. *The software life cycle environment, including the methods and tools to be used for the activities of each software life cycle process has been selected and defined (see* MB.4.4*).*

d. *Additional considerations, such as those discussed in section* MB.12*, have been addressed, if necessary.*

e. *Software development standards,* including the Software Model Standards, *consistent with the system safety objectives for the software to be produced are defined (see* MB.4.5*).*

f. *Software plans that comply with* sections MB.4.3 and MB.11 *have been produced.*

g. *Development and revision of the software plans are coordinated (see* MB.4.3*).*

MB.4.2 Software Planning Process Activities

Effective planning is a determining factor in producing software that satisfies the guidance of this document. Activities for the software planning process include:

a. *The software plans should be developed that provide direction to the personnel performing the software life cycle processes. See also* MB.9.1.

b. *The software development standards to be used for the project should be defined or selected.*

c. *Methods and tools should be chosen that aid error prevention and provide defect detection in the software development processes.*

d. *The software planning process should provide coordination between the software development and integral processes to provide consistency among strategies in the software plans.*

e. *The means should be specified to revise the software plans as a project progresses.*

f. *When multiple-version dissimilar software is used in a system, the software planning process should choose the methods and tools to achieve the dissimilarity necessary to satisfy the system safety objectives.*

g. *For the software planning process to be complete, the software plans and software development standards should be under change control and reviews of them completed (see* MB.4.6*).*

h. *If deactivated code is planned, the software planning process should describe how the deactivation mechanism and deactivated code will be defined and verified to satisfy system safety objectives.*

i. *If user-modifiable software is planned, related processes, tools, environment, and data items substantiating the design (see* MB.5.2.3*) should be specified in the software plans and standards.*

j. *When parameter data items are planned, the following should be addressed:*

1. *The way that parameter data items are used.*

2. *The software level of the parameter data items.*

3. *The processes to develop, verify, and modify parameter data items, and any associated tool qualification.*

4. *Software load control and compatibility.*

k. *The software planning process should address any additional considerations that are applicable.*

l. If software development activities will be performed by a supplier, planning should address supplier oversight.

m. The software planning process should identify the software life cycle data represented by each model, for example, requirements data or design data. Each development model should be categorized as a Specification Model or a Design Model. The software planning process should also identify the methods (that is, review, analysis, and simulation as in section MB.6.8.1) to be used to verify the model. This includes the definition of the model coverage analysis criteria as introduced in section MB.6.7.1 for Design Models. When multiple models are used, the software planning process should address each model.

n. A model is characterized by the modeling technique that is used and by the type of software life cycle data they will represent. Each modeling technique used, and rationale for its suitability to the type of information expressed by the Specification Model or Design Model, should be described in the Software Model Standards (see MB.11.23). Software Model Standards should be developed for each type of the model being developed (that is, a Specification Model or a Design Model).

o. For Design Models, simulation may be used in combination with testing and appropriate analysis to achieve objectives related to the verification of the Executable Object Code. In that case, the planning process should address the specific constraints defined in section MB.6.8.2.

Other software life cycle processes may begin before completion of the software planning process if transition criteria for the specific process activity are satisfied.

MB.4.3 Software Plans

The software plans define the means of satisfying the objectives of this document. They specify the organizations that will perform those activities. The software plans are:

- *The Plan for Software Aspects of Certification (see* MB.11.1*) serves as the primary means for communicating the proposed development methods to the certification authority for agreement, and defines the means of compliance with this document.*
- *The Software Development Plan (see* MB.11.2*) defines the software life cycle(s), software development environment, and the means by which the software development process objectives will be satisfied.*
- *The Software Verification Plan (see* MB.11.3*) defines the means by which the software verification process objectives will be satisfied.*
- *The Software Configuration Management Plan (see* MB.11.4*) defines the means by which the software configuration management process objectives will be satisfied.*
- *The Software Quality Assurance Plan (see* DO-178C section *11.5) defines the means by which the software quality assurance process objectives will be satisfied.*

Activities for the software plans include:

a. The software plans should comply with this document.

b. The software plans should define the transition criteria for software life cycle processes by specifying:

1. *The inputs to the process, including feedback from other processes.*

2. *Any integral process activities that may be required to act on these inputs.*

3. *Availability of tools, methods, plans, and procedures.*

c. *The software plans should state the procedures to be used to implement software changes prior to use on a certified product. Such changes may be as a result of feedback from other processes and may cause a change to the software plans.*

MB.4.4 Software Life Cycle Environment Planning

Planning for the software life cycle environment defines the methods, tools, procedures, programming languages and hardware that will be used to develop, verify, control, and produce the software life cycle data (see MB.11*) and software product. Examples of how the software environment chosen can have a beneficial effect on the software include enforcing standards, detecting errors, and implementing error prevention and fault tolerance methods. The software life cycle environment is a potential error source that can contribute to failure conditions. Composition of this software life cycle environment may be influenced by the safety-related requirements determined by the system safety assessment process, for example, the use of dissimilar, redundant components.*

The goal of error prevention methods is to avoid errors during the software development processes that might contribute to a failure condition. The basic principle is to choose requirements development and design methods, tools, and programming languages that limit the opportunity for introducing errors, and verification methods that ensure that errors introduced are detected. The goal of fault tolerance methods is to include safety features in the software design or Source Code to ensure that the software will respond correctly to input data errors and prevent output and control errors. The need for error prevention or fault tolerance methods is determined by the system requirements and the system safety assessment process.

The considerations presented above may affect:

a. *The methods and notations used in the software requirements process and software design process.*

b. *The programming language(s) and methods used in the software coding process.*

c. *The software development environment tools.*

d. *The software verification and software configuration management tools.*

e. *The need for tool qualification (see* MB.12.2*).*

MB.4.4.1 Software Development Environment

Section 4.4.1 of DO-178C is unchanged.

MB.4.4.2 Language and Compiler Considerations

Upon successful completion of verification of the software product, the compiler is considered acceptable for that product. For this to be valid, the software verification process needs to consider particular features of the programming language and compiler. The software planning process considers these features when choosing a programming language and planning for verification. Activities include:

a. *Some compilers have features intended to optimize performance of the object code. If the test cases give coverage consistent with the software level, the correctness of the optimization need not be verified. Otherwise, the impact of these features on structural coverage analysis should be determined. Additional information can be found in* DO-178C *section 6.4.4.2.*

b. *To implement certain features, compilers for some languages may produce object code that is not directly traceable to the Source Code, for example, initialization, built-in error detection or exception handling (see* DO-178C section *6.4.4.2.b). The software planning process should provide a means to detect this object code and to ensure verification coverage, and should define the means in the appropriate plan.*

c. *If a new compiler, linkage editor, or loader version is introduced, or compiler options are changed during the software life cycle, previous tests, and coverage analyses may no longer be valid. The verification planning should provide a means of reverification that is consistent with* sections MB.6 and MB.12.1.3.

Note: Although the compiler is considered acceptable once all of the verification objectives are satisfied, the compiler is only considered acceptable for that product and not necessarily for other products.

MB.4.4.3 Software Test Environment

Software test environment planning defines the methods, tools, procedures, and hardware that will be used to test the outputs of the integration process. Testing may be performed using the target computer, a target computer emulator, or a host computer simulator. Activities include:

a. *The emulator or simulator may need to be qualified as described in section* MB.12.2.

b. *The differences between the target computer and the emulator or simulator, and the effects of these differences on the ability to detect errors and verify functionality, should be considered. Detection of those errors should be provided by the software verification process and specified in the Software Verification Plan.*

MB.4.4.4 Simulation Environment

Model simulation environment planning defines the methods, tools, procedures, and operating environment that may be used when performing selected verification activities of the outputs of the model development process using a model simulator. Activities include:

a. Determination of whether the model simulator needs to be qualified should be evaluated according to the criteria given in section MB.12.2.

b. Capabilities and limitations of the model simulator with regards to its intended use and their effects on the ability to detect errors and verify functionality should be addressed. Detection of errors not capable of being found by model simulation should be provided by other software verification process activities and the corresponding method(s) specified in the Software Verification Plan.

c. Verification planning should assess the effects of simulation environment changes and consider re-verification consistent with the guidance of sections MB.6 and MB.12.1.3.

MB.4.5 Software Development Standards

Software development standards define the rules and constraints for the software development processes. The software development standards include the Software Requirements Standards, the Software Design Standards, the Software Code Standards, and the Software Model Standards. *The software verification process uses these standards as a basis for evaluating the compliance of actual outputs of a process with intended outputs. Activities for development of the software standards include:*

a. *The software development standards should comply with* section MB.11.

b. *The software development standards should enable software components of a given software product or related set of products to be uniformly designed and implemented.*

c. *The software development standards should disallow the use of constructs or methods that produce outputs that cannot be verified or that are not compatible with safety-related requirements.*

d. *Robustness should be considered in the software development standards.*

Note 1: In developing standards, consideration can be given to previous experience. Constraints and rules on development, design, and coding methods can be included to control complexity. Defensive programming practices may be considered to improve robustness.

Note 2: If allocated to software by system requirements, practices to detect and control errors in stored data, and refresh and monitor hardware status and configuration may be used to mitigate single event upsets.

MB.4.6 Review of the Software Planning Process

Reviews of the software planning process are conducted to ensure that the software plans and software development standards comply with the guidance of this document and means are provided to execute them. Activities include:

a. *Methods are chosen that enable the objectives of this document to be satisfied.*

b. *Software life cycle processes can be applied consistently.*

c. *Each process produces evidence that its outputs can be traced to their activity and inputs, showing the degree of independence of the activity, the environment, and the methods to be used.*

d. *The outputs of the software planning process are consistent and comply with* section MB.11.

MB.5.0 SOFTWARE DEVELOPMENT PROCESSES

This section discusses the objectives and activities of the software development processes. The software development processes are applied as defined by the software planning process (see MB.4*) and the Software Development Plan (see* MB.11.2*).* Table MB.A-2 of Annex MB.A *is a summary of the objectives and outputs of the software development processes by software level. The software development processes are:*

- *Software requirements process.*
- *Software design process.*
- *Software coding process.*
- *Integration process.*

Software development processes produce one or more levels of software requirements, which may be expressed within models. *High-level requirements are produced directly through analysis of system requirements and system architecture. Usually, these high-level requirements are further developed during the software design process, thus producing one or more successive, lower levels of requirements.* Based on the definition of Specification Model and Design Model, high-level requirements and at least one level of low-level requirements always exist. Therefore, this supplement does not allow a single level of requirements.

The development of software architecture involves decisions made about the structure of the software. During the software design process, the software architecture is defined and low-level requirements are developed. Low-level requirements are software requirements from which Source Code can be directly implemented without further information.

Requirements from which the model is developed should be external to the model. Those requirements should provide details and constraints to enable model development and verification activities.

All objectives and activities related to high-level requirements contained in this section are applicable to the requirements contained in Specification Models and to the requirements from which Design Models are developed.

All objectives and activities related to software architecture and low-level requirements contained in this section are applicable to the software architecture and requirements contained in Design Models.

Note: As stated in DO-178C section 2.6, certification credit may be sought from system life cycle processes for the satisfaction, or partial satisfaction, of the software objectives as defined in this supplement.

A model cannot be classified as both a Specification Model and a Design Model.

Each software development process may produce derived requirements. Some examples of requirements that might be determined to be derived requirements are:

- *The need for interrupt handling software to be developed for the chosen target computer.*

- *The specification of a periodic monitor's iteration rate when not specified by the system requirements allocated to software.*

- *The addition of scaling limits when using fixed point arithmetic.*

High-level requirements may include derived requirements, and low-level requirements may include derived requirements. Thus, Specification Models and Design Models may contain derived requirements. *In order to determine the effects of derived requirements on the system safety assessment and system requirements, all derived requirements should be made available to the system processes including the system safety assessment process.*

Model elements that do not contribute to the representation of any software requirement and are not inputs to a subsequent software development activity may be included in a model (for example, comment elements). These elements will not be implemented in the Executable Object Code and therefore need to be identified.

MB.5.1 Software Requirements Process

Section 5.1 of DO-178C is unchanged.

MB.5.1.1 Software Requirements Process Objectives

The objectives of the software requirements process are:

a. High-level requirements are developed.

b. Derived high-level requirements are defined and provided to the system processes, including the system safety assessment process.

c. In the case where high-level requirements are expressed by a Specification Model, all model elements that do not represent software requirements and are not inputs to a subsequent software development process or activity are identified, for example, a comment block.

Note: Although there is no specific objective to verify that such elements not representing requirements were properly identified, the fulfillment of the objective defined in item c is ensured through the verification objectives of section MB.6.3.1.

MB.5.1.2 Software Requirements Process Activities

Inputs to the software requirements process include the system requirements, the hardware interface and system architecture (if not included in the requirements) from the system life cycle processes, and the Software Development Plan, the Software Requirements Standards, and the Software Model Standards (if Specification Models are used) *from the software planning process. When the planned transition criteria have been satisfied, these inputs are used to develop the high-level requirements.*

The primary output of this process is the Software Requirements Data (see MB.11.9*).*

The software requirements process is complete when its objectives and the objectives of the integral processes associated with it are satisfied. Activities for this process include:

a. The system functional and interface requirements that are allocated to software should be analyzed for ambiguities, inconsistencies, and undefined conditions.

b. Inputs to the software requirements process detected as inadequate or incorrect should be reported as feedback to the input source processes for clarification or correction.

c. Each system requirement that is allocated to software should be specified in the high-level requirements.

d. High-level requirements that address system requirements allocated to software to preclude system hazards should be defined.

e. The high-level requirements should conform to the Software Requirements Standards, and be verifiable and consistent. When high-level requirements are expressed by a Specification Model, this model should conform to the Software Model Standards (see MB.11.23), and be verifiable and consistent.

f. The high-level requirements should be stated in quantitative terms with tolerances where applicable.

g. The high-level requirements should not describe design or verification detail except for specified and justified design constraints.

h. Derived high-level requirements and the reason for their existence should be defined.

i. Derived high-level requirements should be provided to the system processes, including the system safety assessment process.

j. If parameter data items are planned, the high-level requirements should describe how any parameter data item is used by the software. The high-level requirements should also specify their structure, the attributes for each of their data elements, and, when applicable, the value of each element. The values of the parameter data item elements should be consistent with the structure of the parameter data item and the attributes of its data elements.

k. When high-level requirements are expressed by a Specification Model, all model elements should be categorized as described in the Software Model Standards (see MB.11.23) by either one of the following:

1. Model elements that represent high-level requirements, including derived requirements.

2. Model elements that do not represent high-level requirements (for example elements used to introduce comments).

l. When a Design Model is planned, high-level requirements for that model should be adequate to support implementation and verification of the Design Model. In particular, the following apply:

1. High-level requirements should contain the level of detail that enables the development and verification of the Design Model.

2. High-level requirements should provide a functional description of the software functionality.

Note: Functional description provides a structured description of software functionality, including interrelationship and interfaces with the software

environment, without prescribing the software architecture. Functional description is not software architecture; functionality is allocated to software components as part of software architecture design process.

MB.5.2 Software Design Process

Section 5.2 of DO-178C is unchanged.

MB.5.2.1 Software Design Process Objectives

The objectives of the software design process are:

a. The software architecture and low-level requirements are developed from the high-level requirements.

b. Derived low-level requirements are defined and provided to the system processes, including the system safety assessment process.

c. In the case where low-level requirements and/or architecture are expressed by a Design Model, all model elements that do not represent software requirements or software architecture and are not inputs to a subsequent software development process or activity are identified, for example, a comment block.

Note: Although there is no specific objective to verify that such elements not representing requirements and/or software architecture were properly identified, the fulfillment of the objective defined in item c is ensured through the verification objectives of section MB.6.3.2 and/or MB.6.3.3.

MB.5.2.2 Software Design Process Activities

The software design process inputs are the Software Requirements Data, the Software Development Plan, the Software Design Standards, and the Software Model Standards (if Design Models are used). *When the planned transition criteria have been satisfied, the high-level requirements are used in the design process to develop software architecture and low-level requirements. This may involve one or more lower levels of requirements.*

The primary output of the process is the Design Description (see MB.11.10*) which includes the software architecture and the low-level requirements.*

The software design process is complete when its objectives and the objectives of the integral processes associated with it are satisfied. Activities for this process include:

a. *Low-level requirements and software architecture developed during the software design process should conform to the Software Design Standards and be traceable, verifiable, and consistent.* When low-level requirements or software architecture are expressed by a Design Model, this model should conform to the Software Model Standards (see MB.11.23) and be traceable, verifiable, and consistent.

b. Derived low-level requirements and the reason for their existence should be defined and analyzed to ensure that the higher level requirements are not compromised.

c. Software design process activities could introduce possible modes of failure into the software or, conversely, preclude others. The use of partitioning or other architectural means in the software design may alter the software level assignment for some components of the software. In such cases, additional data should be

defined as derived requirements and provided to the system processes, including the system safety assessment process.

d. Interfaces between software components, in the form of data flow and control flow, should be defined to be consistent between the components.

e. Control flow and data flow should be monitored when safety-related requirements dictate, for example, watchdog timers, reasonableness-checks, and cross-channel comparisons.

f. Responses to failure conditions should be consistent with the safety-related requirements.

g. Inadequate or incorrect inputs detected during the software design process should be provided to the system life cycle processes, the software requirements process, or the software planning process as feedback for clarification or correction.

h. When low-level requirements and/or architecture are expressed by a Design Model, all model elements should be categorized as described in the Software Model Standards (see MB.11.23) by either one of the following:

1. Model elements that represent low-level requirements, including derived requirements, or architecture.

2. Model elements that do not represent low-level requirements or architecture (for example, elements used to introduce comments).

Note: The current state of software engineering does not permit a quantitative correlation between complexity and the attainment of system safety objectives. While no objective guidance can be provided, the software design process should avoid introducing complexity because as the complexity of software increases, it becomes more difficult to verify the design and to show that the safety-related requirements are satisfied.

MB.5.2.3 Designing for User-Modifiable Software

User-modifiable software is designed to be modified by its users. A modifiable component is that part of the software that is intended to be changed by the user and a non-modifiable component is that which is not intended to be changed by the user. User-modifiable software may vary in complexity. Examples include a single memory bit used to select one of two equipment options, a table of messages, or a memory area that can be programmed, compiled, and linked for maintenance functions. Software of any level can include a modifiable component.

The activities for user-modifiable software include:

a. The non-modifiable component should be protected from the modifiable component to prevent interference in the safe operation of the non-modifiable component. This protection can be enforced by hardware, by software, by the tools used to make the change, or by a combination of the three. If the protection is provided by software, it should be designed and verified at the same software level as the non-modifiable software. If the protection is provided by a tool, the tool should be categorized and qualified as defined in section MB.12.2.

b. The means provided to change the modifiable component should be shown to be the only means by which the modifiable component can be changed.

MB.5.2.4 Designing for Deactivated Code

Section 5.2.4 of DO-178C is unchanged.

MB.5.3 Software Coding Process

Section 5.3 of DO-178C is unchanged.

MB.5.4 Integration Process

The target computer and the Source Code from the software coding process are used with the compiling, linking and loading data (see MB.11.16*) in the integration process to develop the integrated system or equipment.*

MB.5.4.1 Integration Process Objectives

Section 5.4.1 of DO-178C is unchanged.

MB.5.4.2 Integration Process Activities

Section 5.4.2 of DO-178C is unchanged.

MB.5.5 Software Development Process Traceability

The activities related to traceability in DO-178C section 5.5 are fully applicable.

When using model-based development, identification of requirements as per the method defined in the Software Model Standards (see MB.11.23.e and MB.11.23.f) should be used for bi-directional traceability. Means for this traceability should also be defined in the Software Model Standards (see MB.11.23.e).

The traceability for model-based development is no different than for development using traditional methods per DO-178C. Since functional requirements are implemented using combinations of model elements, these combinations should therefore be used for bi-directional traceability.

MB.6.0 SOFTWARE VERIFICATION PROCESS

This section discusses the objectives and activities of the software verification process. Verification is a technical assessment of the outputs of the software planning process, software development processes, and the software verification process. The software verification process is applied as defined by the software planning process (see MB.4*) and the Software Verification Plan (see* MB.11.3*). See* section MB.4.6 *for the verification of the outputs of planning process.*

Verification is not simply testing. Testing, in general, cannot show the absence of errors. As a result, the following sections use the term "verify" instead of "test" to discuss the software verification process activities, which are typically a combination of reviews, analyses, and tests.

Table MB.A-3 through Table MB.A-7 of Annex MB.A *contain a summary of the objectives and outputs of the software verification process, by software level.*

Note: *For lower software levels, less emphasis is on:*

- *Verification of Source Code.*
- *Verification of low-level requirements.*
- *Verification of the software architecture.*
- *Degree of test coverage.*
- *Control of verification procedures.*
- *Independence of software verification process activities.*
- *Overlapping software verification process activities, that is, multiple verification activities, each of which may be capable of detecting the same class of error.*
- *Robustness testing.*
- *Verification activities with an indirect effect on error prevention or detection, for example, conformance to software development standards.*

All objectives and activities related to high-level requirements contained in this section are applicable to the requirements contained in Specification Models and requirements from which Design Models are developed.

All objectives and activities related to software architecture and low-level requirements contained in this section are applicable to the software architecture and requirements contained in Design Models.

Note: As stated in DO-178C section 2.6, certification credit may be sought from system life cycle processes for the satisfaction, or partial satisfaction, of the software objectives as defined in this supplement.

MB.6.1 Purpose of Software Verification

The purpose of the software verification process is to detect and report errors that may have been introduced during the software development processes. Removal of the errors

is an activity of the software development processes. The software verification process verifies that:

a. *The system requirements allocated to software have been developed into high-level requirements that satisfy those system requirements.*

b. *The high-level requirements have been developed into software architecture and low-level requirements that satisfy the high-level requirements. If one or more levels of software requirements are developed between high-level requirements and low-level requirements, the successive levels of requirements are developed such that each successively lower level satisfies its higher level of requirements.* If the output of the design process is a sequence of Design Models which express lower levels of requirements, each Design Model satisfies the requirements from which it is developed.

c. *The software architecture and low-level requirements have been developed into Source Code that satisfies the low-level requirements and software architecture.*

d. *The Executable Object Code satisfies the software requirements (that is, intended function), and provides confidence in the absence of unintended functionality.*

e. *The Executable Object Code is robust with respect to the software requirements such that it can respond properly to abnormal inputs and conditions.*

f. *The means used to perform this verification are technically correct and complete for the software level.*

MB.6.2 Overview of Software Verification Process Activities

Software verification process objectives are satisfied through a combination of reviews, analyses, the development of test cases and procedures, and the subsequent execution of those test procedures. Reviews and analyses provide an assessment of the accuracy, completeness, and verifiability of the software requirements, software architecture, and Source Code. The development of test cases and procedures may provide further assessment of the internal consistency and completeness of the requirements. The execution of the test procedures provides a demonstration of compliance with the requirements.

The inputs to the software verification process include the system requirements, the software requirements, software architecture, Trace Data, Source Code, Executable Object Code, and the Software Verification Plan.

The outputs of the software verification process are recorded in the Software Verification Cases and Procedures (see MB.11.13*), the Software Verification Results (see* MB.11.14*), and the associated Trace Data (see* MB.11.21*).*

The need for the requirements to be verifiable once they have been implemented in the software may itself impose additional requirements or constraints on the software development processes.

Software verification considerations include:

a. *If the code tested is not identical to the airborne software, those differences should be specified and justified.*

b. When it is not possible to verify specific software requirements by exercising the software in a realistic test environment, other means should be provided and their justification for satisfying the software verification process objectives defined in the Software Verification Plan or Software Verification Results.

c. Deficiencies and errors discovered during the software verification process should be reported to other software life cycle processes for clarification and correction as applicable.

d. Reverification should be conducted following corrective actions and/or changes that could impact the previously verified functionality. Reverification should ensure that the modification has been correctly implemented.

e. Verification independence is achieved when the verification activity is performed by a person(s) other than the developer of the item being verified. A tool may be used to achieve equivalence to the human verification activity. For independence, the person who created a set of low-level requirements-based test cases should not be the same person who developed the associated Source Code from those low-level requirements.

MB.6.3 Software Reviews and Analyses

Reviews and analyses are applied to the outputs of the software development processes. One distinction between reviews and analyses is that analyses provide repeatable evidence of correctness and reviews provide a qualitative assessment of correctness. A review may consist of an inspection of an output of a process guided by a checklist or similar aid. An analysis may examine in detail the functionality, performance, traceability, and safety implications of a software component, and its relationship to other components within the system or equipment.

There may be cases where the verification objectives described in this section cannot be completely satisfied via reviews and analyses alone. In such cases, those verification objectives may be satisfied with additional testing on the software product. For example, a combination of reviews, analyses, and tests may be developed to establish the worst-case execution time or verification of the stack usage.

If simulation is being used to verify the completeness and correctness of the requirements contained in the model, then section MB.6.8.1 applies in addition to all of section MB.6.3.

MB.6.3.1 Reviews and Analyses of High-Level Requirements

These review and analysis activities detect and report requirements errors that may have been introduced during the software requirements process. These reviews and analysis activities confirm that the high-level requirements satisfy these objectives:

a. Compliance with system requirements: The objective is to ensure that the system functions to be performed by the software are defined, that the functional, performance, and safety-related requirements of the system are satisfied by the high-level requirements, and that derived requirements and the reason for their existence are correctly defined.

b. Accuracy and consistency: The objective is to ensure that each high-level requirement is accurate, unambiguous, and sufficiently detailed, and that the requirements do not conflict with each other.

c. *Compatibility with the target computer: The objective is to ensure that no conflicts exist between the high-level requirements and the hardware/software features of the target computer, especially system response times and input/output hardware.*

d. *Verifiability: The objective is to ensure that each high-level requirement can be verified.*

e. *Conformance to standards: The objective is to ensure that the Software Requirements Standards were followed during the software requirements process and that deviations from the standards are justified.* When high-level requirements are expressed by a model, the objective is to ensure that the Software Model Standards were followed during the software requirement process and that deviations from the standards are justified.

f. *Traceability: The objective is to ensure that the functional, performance, and safety-related requirements of the system that are allocated to software were developed into the high-level requirements.*

g. *Algorithm aspects: The objective is to ensure the accuracy and behavior of the proposed algorithms, especially in the area of discontinuities.*

MB.6.3.2 Reviews and Analyses of Low-Level Requirements

These review and analysis activities detect and report requirements errors that may have been introduced during the software design process. These reviews and analysis activities confirm that the low-level requirements satisfy these objectives:

a. *Compliance with high-level requirements: The objective is to ensure that the low-level requirements satisfy the high-level requirements and that derived requirements and the design basis for their existence are correctly defined.* This objective is also to provide confidence in detecting the presence of unintended functionality in the Design Model.

b. *Accuracy and consistency: The objective is to ensure that each low-level requirement is accurate and unambiguous and that the low-level requirements do not conflict with each other.*

c. *Compatibility with the target computer: The objective is to ensure that no conflicts exist between the low-level requirements and the hardware/software features of the target computer, especially the use of resources such as bus loading, system response times, and input/output hardware.*

d. *Verifiability: The objective is to ensure that each low-level requirement can be verified.*

e. *Conformance to standards: The objective is to ensure that the Software Design Standards were followed during the software design process and that deviations from the standards are justified.* When low-level requirements are expressed by a model, the objective is to ensure that the Software Model Standards were followed during the software design process and that deviations from the standards are justified.

f. *Traceability: The objective is to ensure that the high-level requirements and derived requirements were developed into the low-level requirements.*

Note: When high-level requirements are expressed by a Specification Model, this objective also aims at ensuring that no low-level requirement traces to model elements that do not represent high-level requirements (see MB.5.1.1.c).

g. *Algorithm aspects: The objective is to ensure the accuracy and behavior of the proposed algorithms, especially in the area of discontinuities.*

MB.6.3.3 Reviews and Analyses of Software Architecture

These review and analysis activities detect and report errors that may have been introduced during the development of the software architecture. These reviews and analysis activities confirm that the software architecture satisfies these objectives:

a. *Compatibility with the high-level requirements: The objective is to ensure that the software architecture does not conflict with the high-level requirements, especially functions that ensure system integrity, for example, partitioning schemes.*

b. *Consistency: The objective is to ensure that a correct relationship exists between the components of the software architecture. This relationship exists via data flow and control flow. If the interface is to a component of a lower software level, it should also be confirmed that the higher software level component has appropriate protection mechanisms in place to protect itself from potential erroneous inputs from the lower software level component.*

c. *Compatibility with the target computer: The objective is to ensure that no conflicts exist, especially initialization, asynchronous operation, synchronization, and interrupts, between the software architecture and the hardware/software features of the target computer.*

d. *Verifiability: The objective is to ensure that the software architecture can be verified, for example, there are no unbounded recursive algorithms.*

e. *Conformance to standards: The objective is to ensure that the Software Design Standards were followed during the software design process and that deviations to the standards are justified, for example, deviations to complexity restrictions and design constraint rules.* When software architecture is expressed by a model, the objective is to ensure that the Software Model Standards were followed during the software design process and that deviations from the standards are justified.

f. *Partitioning integrity: The objective is to ensure that partitioning breaches are prevented.*

MB.6.3.4 Reviews and Analyses of Source Code

These review and analysis activities detect and report errors that may have been introduced during the software coding process. Primary concerns include correctness of the code with respect to the software requirements and the software architecture, and conformance to the Software Code Standards. These reviews and analysis activities are usually confined to the Source Code and confirm that the Source Code satisfies these objectives:

a. *Compliance with the low-level requirements: The objective is to ensure that the Source Code is accurate and complete with respect to the low-level requirements and that no Source Code implements an undocumented function.*

b. *Compliance with the software architecture: The objective is to ensure that the Source Code matches the data flow and control flow defined in the software architecture.*

c. *Verifiability: The objective is to ensure the Source Code does not contain statements and structures that cannot be verified and that the code does not have to be altered to test it.*

d. *Conformance to standards: The objective is to ensure that the Software Code Standards were followed during the development of the code, for example, complexity restrictions and code constraints. Complexity includes the degree of coupling between software components, the nesting levels for control structures, and the complexity of logical or numeric expressions. This analysis also ensures that deviations to the standards are justified.*

e. *Traceability: The objective is to ensure that the low-level requirements were developed into Source Code.*

Note: When low-level requirements are expressed by a Design Model, this objective also aims at ensuring that no Source Code traces to model elements that do not represent low-level requirements (see MB.5.2.1.c).

f. *Accuracy and consistency: The objective is to determine the correctness and consistency of the Source Code, including stack usage, memory usage, fixed point arithmetic overflow and resolution, floating-point arithmetic, resource contention and limitations, worst-case execution timing, exception handling, use of uninitialized variables, cache management, unused variables, and data corruption due to task or interrupt conflicts. The compiler (including its options), the linker (including its options), and some hardware features may have an impact on the worst-case execution timing and this impact should be assessed.*

MB.6.3.5 Reviews and Analyses of the Outputs of the Integration Process

Section 6.3.5 of DO-178C is unchanged.

MB.6.4 Software Testing

Section 6.4 of DO-178C is unchanged.

If simulation is being used as part of Executable Object Code verification, then section MB.6.8.2 applies in addition to section 6.4 of DO-178C.

MB.6.5 Software Verification Process Traceability

Section 6.5 of DO-178C is unchanged.

MB.6.6 Verification of Parameter Data Items

Section 6.6 of DO-178C is unchanged.

MB.6.7 Model Coverage Analysis for Design Models

This section only applies when Design Models are used in the software development process.

Model coverage analysis determines which requirements expressed by the model were not exercised by verification based on the requirements from which the model was developed. Model coverage analysis is a means to assess thoroughness of the model verification activities.

This analysis supports the detection of unintended functionality (with respect to the requirements from which the model was developed) in the Design Model, where coverage of the requirements from which the model was developed has been achieved by the verification cases.

Model coverage analysis is different than structural coverage analysis and therefore model coverage analysis does not eliminate the need to achieve the objectives of structural coverage analysis per DO-178C section 6.4.4.2.

Model coverage analysis should use the outputs (cases, procedures, and/or results) from one or more verification techniques: simulation, testing, and/or other appropriate techniques. Model coverage analysis does not eliminate the need for traceability analysis between requirements from which the model was developed and the model. Traceability analysis can also support model coverage analysis if the Trace Data used for this purpose are adequate (for example, the granularity of the Trace Data is appropriate) to assess the model coverage analysis criteria, as defined during the planning process.

Since model coverage analysis can be supported by different verification techniques, for simplicity purposes, the text in this section will refer to verification cases as one or more of the following:

- Simulation cases.
- Test cases.
- Cases produced through other verification techniques such as formal methods.

Model coverage analysis activities include:

a. Model coverage analysis should be performed using requirements-based verification cases as described in DO-178C section 6.4.2 using the requirements from which the Design Model was developed.

b. Model coverage analysis should confirm the degree of coverage of the Design Model by the verification cases based on the requirements from which the Design Model was developed in accordance with the defined coverage criteria (see MB.6.7.1). In the event complete model coverage cannot be achieved, model coverage resolution should be performed according to section MB.6.7.2.

c. Model coverage analysis should be supplemented with additional verification cases based on derived requirements contained in the Design Model (see MB.6.7.1).

MB.6.7.1 Model Coverage Analysis Criteria

As the intent of model coverage analysis is to detect unintended function in the Design Model, it is recognized that various methods are available to perform the model coverage analysis. In particular, the model coverage analysis criteria can be defined in multiple ways but should comply with DO-178C sections 6.4.2.1 and 6.4.2.2. Therefore, the criteria should be defined during the planning process and indicated in the Software

Verification Plan. The following shows an example of criteria that may be used to assess model coverage; however, alternative criteria may be proposed.

Table MB.6-1 Model Coverage Criteria Example

Typical Completion Criteria	Satisfy by verification cases and justifications based on requirements from which the Design Model was developed	Satisfy by verification cases and justifications based on requirements contained in the Design Model
Coverage of all characteristics of the functionality in context, for example watchdog function triggered (See Note)	**Recommended**	
For state machines: coverage of all the transitions (See Note)	**Recommended**	
For logic equations: coverage of all the decisions (See Note)	**Recommended**	
Coverage of all equivalence classes and boundary/singular values for numeric data (See Note)	**Recommended**	**Alternatively**
Coverage of all derived requirements (not traceable to higher-level requirements)		**Recommended**

Note: These criteria are compatible with DO-178C sections 6.4.2.1 and 6.4.2.2.

MB.6.7.2 Model Coverage Analysis Resolution

Possible causes of verification deficiencies revealed by model coverage analysis, and the additional activities required to resolve the deficiencies, include:

a. Shortcomings in requirements-based verification cases or procedures: The verification cases should be supplemented or verification procedures changed to provide the missing coverage. The method(s) used to perform the coverage analyses based on the requirements from which the Design Model was developed may need to be reviewed.

b. Inadequacies or shortcomings in requirements from which the Design Model was developed: The requirements from which the Design Model was developed should be modified and additional verification cases developed and verification procedures executed.

c. Derived requirements expressed by the Design Model: Appropriate verification cases should exist or be developed for the derived requirements and verification procedures should be executed to provide the missing coverage. Previously unidentified derived requirements should be identified, justified, and provided to the system processes, including system safety assessment process.

d. Deactivated functionality expressed by the Design Model: Deactivated functionality expressed by the Design Model should be justified. For deactivated functionality expressed by a Design Model that is not intended to be realized in any configuration used within an aircraft or engine, a combination of analysis, simulation and testing

should show that its realization is prevented, isolated, or eliminated. For deactivated functionality expressed by a Design Model that is only intended to be realized in certain approved configurations used within an aircraft or engine, the operational configuration needed for normal realization of these requirements should be established and additional verification cases and verification procedures developed to satisfy the required coverage objectives.

e. Unintended functionality expressed by a Design Model: Unintended functionality indicates an error is present and the functionality should be removed from the Design Model.

MB.6.8 Model Simulation

The goal of this section is to provide guidance, including limitations, to allow the use of simulation without compromising the rigor appropriate for the software level. The use of model simulation in compliance with this supplement does not alleviate the guidance of DO-178C sections 6.3 and 6.4. When simulation is used, all objectives of high-level requirements verification, software design verification, Executable Object Code verification, and software testing verification, should still be satisfied (see MB.6.3.1, MB.6.3.2, MB.6.3.3, and MB.6.4). Model simulation may support the achievement of some of these objectives. In this case, it is necessary to ensure that model simulation is planned, developed, and performed completely and correctly to support satisfaction of those specific objectives.

For Specification Models or Design Models, simulation may be used in combination with reviews and analysis of requirements and architecture to satisfy some objectives of sections MB.6.3.1, MB.6.3.2, and MB.6.3.3. In this case, guidance on the verification of a model by means of simulation in section MB.6.8.1 should be followed.

For Design Models, simulation may be used in combination with testing and appropriate coverage analysis to satisfy objectives related to the verification of the Executable Object Code. As simulation may involve different object code and a different environment than the target application, simulation alone cannot be used to satisfy objectives related to verification of the Executable Object Code. Guidance on the verification of Executable Object Code by means of simulation in section MB.6.8.2 should be followed. However, if simulation cases are run in the target computer environment using the Executable Object Code, then they are also considered test cases and DO-178C section 6.4 applies.

MB.6.8.1 Model Simulation for Verification of the Model

The main purpose of simulation is to provide repeatable evidence of compliance of the model to the requirements from which the model was developed, satisfying the following objectives:

- Compliance to system requirements for Specification Models (see MB.6.3.1.a).
- Compliance to software high-level requirements for Design Models containing low-level requirements (see MB.6.3.2.a).
- Compliance to software high-level requirements for Design Models containing software architecture (see MB.6.3.3.a).

Several other objectives may be satisfied by simulation for Specification Models and Design Models such as:

- Accuracy and consistency (see MB.6.3.1.b and MB.6.3.2.b).
- Verifiability (see MB.6.3.1.d and MB.6.3.2.d).
- Algorithm aspects (see MB.6.3.1.g and MB.6.3.2.g).

In addition, simulation may provide evidence for Design Models containing software architecture of the following:

- Consistency (see MB.6.3.3.b).
- Verifiability (see MB.6.3.3.d).

However, simulation cannot be used to satisfy the following objectives:

- Compatibility with the target computer (see MB.6.3.1.c, MB.6.3.2.c, and MB.6.3.3.c).
- Conformance to standards (see MB.6.3.1.e, MB.6.3.2.e, and MB.6.3.3.e).
- Traceability (see MB.6.3.1.f and MB.6.3.2.f).
- Partitioning integrity (see MB.6.3.3.f).

Note: Model simulation activity is not limited to the execution of the simulation cases and procedures but also encompasses the review/analysis of the requirements in order to determine which objectives can be supported by simulation as well as the review/analysis of the simulation results to confirm the objectives have been satisfied (see MB.6.8.3.2.c).

When certification credit is sought from model simulation for the satisfaction of any objectives of high-level requirements verification or software design verification (see MB.6.3.1, MB.6.3.2, and MB.6.3.3), the applicant should perform the following activities:

a. Determine what reviews and analyses objectives are planned to be better achieved by simulation; all other objectives should be satisfied by reviews and analyses as described in section MB.6.3.

b. Justify, in detail, how that simulation activity completely satisfies the specific review or analysis objectives.

c. Perform an analysis to determine that:

   1. Simulation cases exist for each requirement from which the model was developed.

   2. Simulation cases satisfy the criteria of normal range and robustness as defined in DO-178C section 6.4.2.

   Note: This analysis may reveal the need for additional simulation cases.

In addition, whenever simulation is used, accurate simulation cases and procedures should be prepared as defined in section MB.6.8.3 and the simulation environment should be planned as defined in section MB.4.4.4.

MB.6.8.2 Model Simulation for Verification of the Executable Object Code

Verification of the Executable Object Code is primarily performed by testing. This can be partially assisted by a combination of model simulation and specific analyses as described below. This combination can be used to partially satisfy the following software testing and test coverage objectives:

- Executable Object Code complies with the high-level requirements (see DO-178C section 6.4.a).
- Executable Object Code is robust with the high-level requirements (see DO-178C section 6.4.b).
- Test coverage of high-level requirements is achieved (see DO-178C section 6.4.4.a).
- Test coverage of software structure to the appropriate coverage criteria is achieved (see DO-178C section 6.4.4.c).
- Test coverage of software structure, both data coupling and control coupling is achieved (see DO-178C section 6.4.4.d).

But specific tests should still be performed in the target computer environment, since some errors may be detectable only in this environment.

The following software testing and test coverage objectives cannot be satisfied by model simulation since simulation cases should be based on the requirements from which the Design Model is developed:

- Executable Object Code complies with the low-level requirements (see DO-178C section 6.4.c).
- Executable Object Code is robust with the low-level requirements (see DO-178C section 6.4.d).
- Test coverage of low-level requirements is achieved (see DO-178C section 6.4.4.b).

The software testing objective, Executable Object Code is compatible with the target computer, cannot be satisfied by means of simulation (see DO-178C 6.4.e). In particular, tests dedicated to hardware/software integration verification always need to be performed in the target computer environment as defined in DO-178C section 6.4.3.a.

The following aspects apply when model simulation is used for Executable Object Code verification:

a. When certification credit is sought from model simulation to partially satisfy software testing objectives and test coverage regarding high-level requirements (see DO-178C 6.4.a, 6.4.b, and 6.4.4.a), then these activities include:

   1. Ensuring that the same Design Model is used for simulation as is used for the production of the Source Code and/or Executable Object Code.

   2. Determining what software testing and test coverage objectives and which specific high-level requirements are planned to be satisfied by simulation; all other software testing and test coverage objectives should be satisfied by testing as described in DO-178C section 6.4. Typical errors listed in DO-178C section

6.4.3.a cannot be detected by model simulation since they require the target computer hardware.

Typical errors consistent with DO-178C sections 6.4.3.b and 6.4.3.c that may be revealed by model simulation include:

- Inadequate end-to-end numerical resolution.
- Incorrect sequencing of events and operations.
- Failure of an algorithm to satisfy a software requirement.
- Incorrect loop operations.
- Incorrect logic decisions.
- Failure to process correctly legitimate combinations of input conditions.
- Incorrect responses to missing or corrupted input data.
- Incorrect computation sequence.
- Inadequate algorithm precision, accuracy, or performance.
- Incorrect state transitions.

Model simulation cannot be used to detect errors related to the target computer hardware, for example:

- Incorrect handling of exceptions, such as arithmetic faults or violations of array limits.
- Data corruption, especially global data.
- Timing related requirements and performance.
- Hardware resource related performance.
- Hardware monitoring requirements (for example, built-in test).

3. Justifying, in detail, how the simulation activity satisfies the specific software testing and test coverage objectives identified in item 2 above. To this purpose, an analysis should provide compelling evidence that the simulation approach provides equivalent defect detection and removal as testing of the Executable Object Code in compliance with DO-178C section 6.4.3. This analysis should:

   i. Verify how representative the model simulator environment is compared to the target computer environment, addressing processor differences, if any. Topics such as floating point precision, integer word sizes, math libraries, set of instructions, etc. should be addressed, depending on the certification credit sought. The analysis should be supported by the execution of a representative set of verification procedures on both environments. In addition, the analysis may rely upon confidence gained from features such as target system's hardware abstraction layer and/or target computer system's operating system capabilities.

ii. Address the equivalence and differences between the simulation executable object code and the target Executable Object Code. These differences can be the result of:

- Different compiling and linking process: The differences to be considered are typically compiler options and/or optimizations. These differences should be analyzed with respect to the software testing objectives identified in item 2 above. The analysis should be supported by the execution of a representative set of verification procedures on both environments. Considerations defined in section MB.4.4.2.a should be addressed during this analysis.

- Source Code differences: Though the source code used for simulation is expected to be the same Source Code used to produce the target Executable Object Code (for example, the same autocode generator is used for both), any applicable differences to improve testability (for example, source code or model element library instrumentation) should be justified regarding their effects on the produced executable object code and subsequent achievement of the software testing objectives identified in item 2 above.

Note: The two items above are not mutually exclusive.

b. When certification credit is sought from model simulation, simulation results may partially support the test coverage objectives of the software structure (see DO-178C 6.4.4.c and 6.4.4.d) only if the source code used for simulation is identical to the Source Code used to produce the target Executable Object Code.

In addition, whenever simulation is used, simulation cases and procedures should be prepared as defined in section MB.6.8.3 and an appropriate representative simulation environment for its intended use should be planned as described in section MB.4.4.4.

MB.6.8.3 Simulation Cases, Procedures, and Results

Simulation consists of the development of simulation cases and procedures based on requirements from which the model was developed, and the subsequent execution of those simulation procedures in a model simulation environment.

When simulation is used as part of the verification activities, the means of developing the code and the means of verifying the code (for example, automatic generation of test cases) should be independent to satisfy independent aspects of Annex MB.A Tables.

MB.6.8.3.1 Development of Simulation Cases, Procedures, and Results

Simulation case selection strategy should address the same considerations as in DO-178C section 6.4.2 with regards to normal range and robustness test cases.

MB.6.8.3.2 Reviews and Analyses of Simulation Cases, Procedures, and Results

When simulation is used as a means to satisfy objective(s) of this supplement, then the simulation cases, procedures, and results should be reviewed and/or analyzed to ensure that the simulation was performed accurately and completely with respect to those objectives.

a. Simulation cases: the objectives related to the verification of test cases as presented in DO-178C sections 6.4.4.a and 6.4.4.b are also applicable for simulation cases.

b. Simulation procedures: the objective is to verify that the simulation cases, including expected results, were correctly developed into simulation procedures.

c. Simulation results: the objective is to ensure that the simulation results are correct and that discrepancies between actual and expected results are explained.

MB.7.0 SOFTWARE CONFIGURATION MANAGEMENT PROCESS

This section discusses the objectives and activities of the software configuration management (SCM) process. The SCM process is applied as defined by the software planning process (see MB.4*) and the Software Configuration Management Plan (see* MB.11.4*). Outputs of the SCM process are recorded in Software Configuration Management Records (see* DO-178C section *11.18) or in other software life cycle data.*

The SCM process, working in cooperation with the other software life cycle processes, assists in:

a. *Providing a defined and controlled configuration of the software throughout the software life cycle.*

b. *Providing the ability to consistently replicate the Executable Object Code and Parameter Data Item Files, if any, for software manufacture or to regenerate it in case of a need for investigation or modification.*

c. *Providing control of process inputs and outputs during the software life cycle that ensures consistency and repeatability of process activities.*

d. *Providing a known point for review, assessing status, and change control by control of configuration items and the establishment of baselines.*

e. *Providing controls that ensure problems receive attention and changes are recorded, approved, and implemented.*

f. *Providing evidence of approval of the software by control of the outputs of the software life cycle processes.*

g. *Assessing the software product compliance with requirements.*

h. *Ensuring that secure physical archiving, recovery, and control are maintained for the configuration items.*

MB.7.1 Software Configuration Management Process Objectives

The SCM process objectives are:

a. *Each configuration item and its successive versions are labeled unambiguously so that a basis is established for the control and reference of configuration items.*

b. *Baselines are defined for further software life cycle process activity and allow reference to, control of, and traceability between, configuration items.*

c. *The problem reporting process records process non-compliance with software plans and standards, records deficiencies of outputs of software life cycle processes, records anomalous behavior of software products, and ensures resolution of these problems.*

d. *Change control provides for recording, evaluation, resolution, and approval of changes throughout the software life cycle.*

e. *Change review ensures problems and changes are assessed, approved, or disapproved, approved changes are implemented, and feedback is provided to*

affected processes through problem reporting and change control methods defined during the software planning process.

f. Status accounting provides data for the configuration management of software life cycle processes with respect to configuration identification, baselines, Problem Reports, and change control.

g. Archival and retrieval ensures that the software life cycle data associated with the software product can be retrieved in case of a need to duplicate, regenerate, retest or modify the software product. The objective of the release activity is to ensure that only authorized software is used, especially for software manufacturing, in addition to being archived and retrievable.

h. Software load control ensures that the Executable Object Code and Parameter Data Item Files, if any, are loaded into the system or equipment with appropriate safeguards.

i. Software life cycle environment control ensures that the tools used to produce the software are identified, controlled, and retrievable.

The objectives for SCM are independent of software level. However, two categories of software life cycle data may exist based on the SCM controls applied to the data (see MB.7.3*).*

Table MB.A-8 of Annex MB.A *is a summary of the objectives and outputs of the SCM process.*

MB.7.2 Software Configuration Management Process Activities

Section 7.2 of DO-178C is unchanged.

MB.7.2.1 Configuration Identification

Section 7.2.1 of DO-178C is unchanged.

MB.7.2.2 Baselines and Traceability

Activities include:

a. Baselines should be established for configuration items used for certification credit. Intermediate baselines may be established to aid in controlling software life cycle process activities.

b. A software product baseline should be established for the software product and defined in the Software Configuration Index (see MB.11.16*).*

Note: User-modifiable software is not included in the software product baseline, except for its associated protection and boundary components. Therefore, modifications may be made to user-modifiable software without affecting the configuration identification of the software product baseline.

c. Baselines should be established in both *controlled software libraries* and controlled model element libraries*, whether physical, electronic, or other, to ensure their integrity. Once a baseline is established, it should be protected from change.*

d. Change control activities should be followed to develop a derivative baseline from an established baseline.

e. A baseline should be traceable to the baseline from which it was derived, if certification credit is sought for software life cycle process activities or data associated with the development of the previous baseline.

f. A configuration item should be traceable to the configuration item from which it was derived, if certification credit is sought for software life cycle process activities or data associated with the development of the previous configuration item.

g. A baseline or configuration item should be traceable either to the output it identifies or to the process with which it is associated.

MB.7.2.3 Problem Reporting, Tracking, and Corrective Action

Section 7.2.3 of DO-178C is unchanged.

MB.7.2.4 Change Control

Section 7.2.4 of DO-178C is unchanged.

MB.7.2.5 Change Review

Section 7.2.5 of DO-178C is unchanged.

MB.7.2.6 Configuration Status Accounting

Section 7.2.6 of DO-178C is unchanged.

MB.7.2.7 Archive, Retrieval, and Release

Section 7.2.7 of DO-178C is unchanged.

MB.7.3 Data Control Categories

Software life cycle data can be assigned to one of two configuration management control categories: Control Category 1 (CC1) and Control Category 2 (CC2). Table MB.7-1 *defines the set of SCM process activities associated with each control category, where ● indicates the minimum activities that apply for software life cycle data of that category. CC2 activities are a subset of the CC1 activities.*

The Annex MB.A tables *specify the control category by software level for the software life cycle data items.*

Table MB.7-1 *Scm Process Activities Associated With Cc1 And Cc2 Data*

SCM Process Activity	*Reference*	*CC1*	*CC2*
Configuration Identification	*7.2.1*	●	●
Baselines	MB.7.2.2.a MB.7.2.2.b MB.7.2.2.c MB.7.2.2.d MB.7.2.2.e	●	
Traceability	MB.7.2.2.f MB.7.2.2.g	●	●
Problem Reporting	*7.2.3*	●	
Change Control - integrity and identification	*7.2.4.a* *7.2.4.b*	●	●
Change Control - tracking	*7.2.4.c* *7.2.4.d* *.7.2.4.e*	●	
Change Review	*7.2.5*	●	
Configuration Status Accounting	*7.2.6*	●	
Retrieval	*7.2.7.a*	●	●
Protection against Unauthorized Changes	*7.2.7.b.1*	●	●
Media Selection, Refreshing, Duplication	*7.2.7.b.2* *7.2.7.b.3* *7.2.7.b.4* *7.2.7.c*	●	
Release	*7.2.7.d*	●	
Data Retention	*7.2.7.e*	●	●

MB.7.4 Software Load Control

Section 7.4 of DO-178C is unchanged.

MB.7.5 Software Life Cycle Environment Control

The software life cycle environment tools are defined by the software planning process and identified in the Software Life Cycle Environment Configuration Index (see MB.11.15*). Activities include:*

a. *Configuration identification should be established for the Executable Object Code, or equivalent, of the tools used to develop, control, build, verify, and load the software.*

b. *The SCM process for controlling qualified tools should comply with the objectives associated with Control Category 1 or Control Category 2 data (see* MB.7.3*), according to the guidance provided by* DO-178C *section 12.2.3.*

c. *Unless section* MB.7.5.b *applies, the SCM process for controlling the Executable Object Code, or equivalent, for tools used to build and load the software (for example, compilers, assemblers, and linkage editors) should comply with the objectives associated with Control Category 2 data, as a minimum.*

MB.8.0 SOFTWARE QUALITY ASSURANCE PROCESS

Section 8.0 of DO-178C is unchanged.

MB.8.1 Software Quality Assurance Process Objectives

The SQA process objectives provide confidence that the software life cycle processes produce software that conforms to its requirements by assuring that these processes are performed in compliance with the approved software plans and standards.

The objectives of the SQA process are to obtain assurance that:

a. *Software plans and standards are developed and reviewed for compliance with this document and for consistency.*

b. *Software life cycle processes, including those of suppliers, comply with approved software plans and standards.*

c. *The transition criteria for the software life cycle processes are satisfied.*

d. *A conformity review of the software product is conducted.*

Table MB.A-9 of Annex MB.A *is a summary of the objectives and outputs of the SQA process.*

MB.8.2 Software Quality Assurance Process Activities

Activities for satisfying the SQA process objectives include:

a. *The SQA process should take an active role in the activities of the software life cycle processes, and have those performing the SQA process enabled with the authority, responsibility, and independence to ensure that the SQA process objectives are satisfied.*

b. *The SQA process should provide assurance that software plans and standards are developed and reviewed for compliance with this document and for consistency.*

c. *The SQA process should provide assurance that the software life cycle processes comply with the approved software plans and standards.*

d. *The SQA process should include audits of the software life cycle processes during the software life cycle to obtain assurance that:*

   1. *Software plans are available as specified in section* MB.4.2.

   2. *Deviations from the software plans and standards are detected, recorded, evaluated, tracked, and resolved.*

      Note: It is generally accepted that early detection of process deviations assists efficient achievement of software life cycle process objectives.

   3. *Approved deviations are recorded.*

   4. *The software development environment has been provided as specified in the software plans.*

5. *The problem reporting, tracking, and corrective action process activities comply with the Software Configuration Management Plan.*

6. *Inputs provided to the software life cycle processes by the system processes, including the system safety assessment process, have been addressed.*

Note: Monitoring of the activities of software life cycle processes may be performed to provide assurance that the activities are under control.

e. The SQA process should provide assurance that the transition criteria for the software life cycle processes have been satisfied in compliance with the approved software plans.

f. The SQA process should provide assurance that software life cycle data is controlled in accordance with the control categories as defined in section MB.7.3 *and the tables of* Annex MB.A.

g. Prior to the delivery of software products submitted as part of a certification application, a software conformity review should be conducted.

h. The SQA process should produce records of the SQA process activities (see DO-178C section *11.19), including audit results and evidence of completion of the software conformity review for each software product submitted as part of certification application.*

i. The SQA process should provide assurance that supplier processes and outputs comply with approved software plans and standards.

MB.8.3 Software Conformity Review

Section 8.3 of DO-178C is unchanged.

MB.9.0 CERTIFICATION LIAISON PROCESS

The objectives of the certification liaison process are to:

a. *Establish communication and understanding between the applicant and the certification authority throughout the software life cycle to assist the certification process.*

b. *Gain agreement on the means of compliance through approval of the Plan for Software Aspects of Certification.*

c. *Provide compliance substantiation.*

The certification liaison process is applied as defined by the software planning process (see MB.4*) and the Plan for Software Aspects of Certification (see* MB.11.1*).* Table MB.A-10 of Annex MB.A *is a summary of the objectives and outputs of this process.*

MB.9.1 Means of Compliance and Planning

The applicant proposes a means of compliance that defines how the development of the airborne system or equipment will satisfy the certification basis. The Plan for Software Aspects of Certification (see MB.11.1*) defines the software aspects of the airborne system or equipment within the context of the proposed means of compliance. This plan also states the software level(s) as determined by the system safety assessment process.*

Activities include:

a. *Submitting the Plan for Software Aspects of Certification and other requested data to the certification authority for review.*

b. *Resolving issues identified by the certification authority concerning the planning for the software aspects of certification.*

c. *Obtaining agreement with the certification authority on the Plan for Software Aspects of Certification.*

MB.9.2 Compliance Substantiation

The applicant provides evidence that the software life cycle processes satisfy the software plans, by making software life cycle data available to the certification authority for review. Certification authority reviews may take place at various facilities. For example, the reviews may take place at the applicant's facilities, the applicant's supplier's facilities, or at the certification authority's facilities. This may involve discussions with the applicant or its suppliers. The applicant arranges these reviews of the activities of the software life cycle processes and makes software life cycle data available as needed.

Activities include:

a. *Resolving issues raised by the certification authority as a result of its reviews.*

b. *Submitting the Software Accomplishment Summary (see* DO-178C section *11.20) and Software Configuration Index (see* MB.11.16*) to the certification authority.*

c. *Submitting or making available other data or evidence of compliance requested by the certification authority.*

MB.9.3 Minimum Software Life Cycle Data Submitted to Certification Authority

Section 9.3 of DO-178C is unchanged.

MB.9.4 Software Life Cycle Data Related to Type Design

Section 9.4 of DO-178C is unchanged.

MB.10.0 OVERVIEW OF CERTIFICATION PROCESS

This section is an overview of the certification process with respect to software aspects of airborne systems and equipment, and is provided for information purposes only.

The airborne community and certification authorities use several terms related to aircraft approval for flight with its associated equipment. The terms used are "certification", "approval", and, with respect to tools, "qualification".

"Certification" applies to aircraft, engines, or propellers; and, in respect of some certification authorities, auxiliary power units. The certification authorities consider the software as part of the airborne system or equipment installed on the certified product; that is, the certification authorities do not certify the software as a unique, stand-alone product.

Systems and equipment, including embedded software, should be "approved" in order to be accepted as a part of a certification. Approval by the certification authorities is given dependent upon a successful demonstration or by review of the products of the software life cycle. Any such approval currently has significance only within the context of a specific certification.

Tool "qualification" is discussed in section MB.12.2.

MB.10.1 Certification Basis

Section 10.1 of DO-178C is unchanged.

MB.10.2 Software Aspects of Certification

Section 10.2 of DO-178C is unchanged.

MB.10.3 Compliance Determination

Prior to certification, the certification authority determines that the product to be certified, including the software aspects of its systems or equipment, complies with the certification basis. For the software, this is accomplished by reviewing the Software Accomplishment Summary and evidence of compliance. The certification authority uses the Software Accomplishment Summary as an overview for the software aspects of certification.

The certification authority may review at its discretion the software life cycle processes and their outputs during the software life cycle, as discussed in section MB.9.2.

This Page Intentionally Left Blank

MB.11.0 SOFTWARE LIFE CYCLE DATA

Data is produced during the software life cycle to plan, direct, explain, define, record, or provide evidence of activities. This data enables the software life cycle processes, system or equipment certification, and post-certification modification of the software product. This section discusses the characteristics, form, configuration management controls, and content of the software life cycle data.

a. *Characteristics: Software life cycle data should be:*

   1. *Unambiguous: Information is unambiguous if it is written in terms which only allow a single interpretation, aided, if necessary, by a definition.*

   2. *Complete: Information is complete when it includes necessary and relevant requirements and/or descriptive material; responses are defined for the range of valid input data; figures used are labeled; and terms and units of measure are defined.*

   3. *Verifiable: Information is verifiable if it can be checked for correctness by a person or tool.*

   4. *Consistent: Information is consistent if there are no conflicts within it.*

   5. *Modifiable: Information is modifiable if it is structured and has a style such that changes can be made completely, consistently, and correctly while retaining the structure.*

   6. *Traceable: Information is traceable if the origin of its components can be determined.*

b. *Form: The form of the software life cycle data should provide for the efficient retrieval and review of software life cycle data throughout the service life of the airborne system or equipment. The data and the specific form of the data should be specified in the Plan for Software Aspects of Certification.*

   Note 1: The software life cycle data may be held in a number of forms (for example, held electronically or in printed form).

   Note 2: The applicant may package software life cycle data items in any manner the applicant finds convenient (for example, as individual data items or as a combined data item).

   Note 3: The Plan for Software Aspects of Certification and the Software Accomplishment Summary may be required as separate documents by some certification authorities.

   Note 4: The term "data" refers to evidence and other information and does not imply the format such data should take.

c. *Configuration management controls: Software life cycle data can be placed in one of two categories associated with the software configuration management controls applied: CC1 and CC2 (see* MB.7.3*). The minimum control category assigned to each data item, and its variation by software level is specified in the tables of* Annex MB.A. *If additional data items than those described herein are produced as evidence to aid the certification process, they should be, as a minimum, under CC2 controls.*

d. *Content: The software life cycle data descriptions provided in the following sections identify the data that is generally produced by the software life cycle. The descriptions are not intended to describe all data that may be necessary to develop a software product, and are not intended to imply a particular data packaging method or organization of the data within a package. The descriptions of the content of software life cycle data items provided herein are not all encompassing and should be read in conjunction with the body of this document and adapted to the needs of the applicant.*

MB.11.1 Plan for Software Aspects of Certification

The Plan for Software Aspects of Certification (PSAC) is the primary means used by the certification authority for determining whether an applicant is proposing a software life cycle that is commensurate with the rigor required for the level of software being developed. This plan should include:

a. *System overview: This section provides an overview of the system, including a description of its functions and their allocation to the hardware and software, the architecture, processor(s) used, hardware/software interfaces, and safety features.*

b. *Software overview: This section briefly describes the software functions with emphasis on the proposed safety and partitioning concepts. Examples include resource sharing, redundancy, fault tolerance, mitigation of single event upset, and timing and scheduling strategies.*

c. *Certification considerations: This section provides a summary of the certification basis, including the means of compliance, as relating to the software aspects of certification. This section also states the proposed software level(s) and summarizes the justification provided by the system safety assessment process, including potential software contributions to failure conditions.*

d. *Software life cycle: This section defines the software life cycle to be used and includes a summary of each of the software life cycle processes for which detailed information is defined in their respective software plans.* The software life cycle processes should accommodate the technology from the applicable supplements. *The summary explains how the objectives of each software life cycle process* (DO-178C objectives as well as the applicable supplement objectives) *will be satisfied, and specifies the organizations to be involved, the organizational responsibilities, and the system life cycle processes and certification liaison process responsibilities.* This summary should include processes and activities performed in accordance with any supplement. If there are any conflicts between objectives either of DO-178C and/or supplements, an approach to resolve the conflicts should be included.

e. *Software life cycle data: This section specifies the software life cycle data that will be produced and controlled by the software life cycle processes. This section also describes the relationship of the data to each other or to other data defining the system,* including data produced in accordance with any supplement used, *the software life cycle data to be submitted to the certification authority, the form of the data, and the means by which the data will be made available to the certification authority.*

f. Schedule: *This section describes the means the applicant will use to provide the certification authority with visibility of the activities of the software life cycle processes so reviews can be planned.*

g. Additional considerations: *This section describes specific considerations that may affect the certification process. Examples include alternative methods of compliance, tool qualification, previously developed software, option-selectable software, user-modifiable software, deactivated code, COTS software, field-loadable software, parameter data items, multiple-version dissimilar software, and product service history.*

h. Supplier oversight: *This section describes the means of ensuring that supplier processes and outputs will comply with approved software plans and standards.*

MB.11.2 Software Development Plan

The Software Development Plan (SDP) is a description of the software development procedures and software life cycle(s) to be used to satisfy the software development process objectives. It may be included in the Plan for Software Aspects of Certification. This plan should include:

a. Standards: *Identification of the Software Requirements Standards, Software Design Standards, Software Code Standards,* and Software Model Standards *for the project. Also, references to the standards for previously developed software, including COTS software, if those standards are different.*

b. Software life cycle: *A description of the software life cycle processes to be used to form the specific software life cycle(s) to be used on the project, including the transition criteria for the software development processes. This description is distinct from the summary provided in the Plan for Software Aspects of Certification, in that it provides the detail necessary to ensure proper implementation of the software life cycle processes.*

c. Software development environment: *A statement of the chosen software development environment in terms of hardware and software, including:*

1. *The requirements development method(s) and tools to be used.*
2. *The design method(s) and tools to be used.*
3. *The coding method(s), programming language(s), coding tool(s) to be used, and when applicable, options and constraints of autocode generators.*
4. *The compilers, linkage editors, and loaders to be used.*
5. *The hardware platforms for the tools to be used.*
6. The modeling method(s), modeling language(s), and modeling tool(s) to be use

MB.11.3 Software Verification Plan

The Software Verification Plan (SVP) is a description of the verification procedures to be used to satisfy the software verification process objectives. These procedures may vary by software level as defined in the tables of Annex MB.A. *This plan should include:*

a. *Organization: Organizational responsibilities within the software verification process and interfaces with the other software life cycle processes.*

b. *Independence: A description of the methods for establishing verification independence, when required.*

c. *Verification methods: A description of the verification methods to be used for each activity of the software verification process.*

1. *Review methods, including checklists or other aids.*

2. *Analysis methods, including traceability and coverage analysis.* Model traceability analysis, model coverage criteria, and model coverage analysis should be addressed.

3. *Testing methods, including the method for selecting test cases, the test procedures to be used, and the test data to be produced.*

4. Model simulation methods, including the method for selecting the simulation cases, the simulation procedures to be used, and the simulation data to be produced. This description of model simulation methods should address the specific constraints defined in sections MB.6.8.1 and MB.6.8.2.

d. *Verification environment: A description of the equipment for testing, the testing and analysis tools, and how to apply these tools and hardware test equipment.* DO-178C *section 4.4.3.b provides guidance on indicating target computer and simulator or emulator differences.* This should also include a description of the model simulation environment and tools, and the guidance for applying these tools.

e. *Transition criteria: The transition criteria for entering the software verification process.*

f. *Partitioning considerations: If partitioning is used, the methods used to verify the integrity of the partitioning.*

g. *Compiler assumptions: A description of the assumptions made by the applicant about the correctness of the compiler, linkage editor, or loader (see* DO-178C section *4.4.2).*

h. *Reverification method: For software modification, a description of the methods for identifying, analyzing and verifying the affected areas of the software and the changed parts of the Executable Object Code.*

i. *Previously developed software: For previously developed software, if the initial compliance baseline for the verification process does not comply with this document, a description of the methods to satisfy the objectives of this document.*

j. *Multiple-version dissimilar software: If multiple-version dissimilar software is used, a description of the software verification process activities (see* DO-178C section *12.3.2).*

MB.11.4 Software Configuration Management Plan

The Software Configuration Management Plan establishes the methods to be used to achieve the objectives of the SCM process throughout the software life cycle. This plan should include:

a. *Environment: A description of the SCM environment to be used, including procedures, tools, methods, standards, organizational responsibilities, and interfaces.*

b. *Activities: A description of the SCM process activities in the software life cycle:*

1. *Configuration identification: Items to be identified, when they will be identified, the identification methods for software life cycle data (for example, part numbering), and the relationship of software identification and the system or equipment identification.*

2. *Baselines and traceability: The means of establishing baselines, what baselines will be established, when these baselines will be established, the software library* and model element library *controls, and the configuration item and baseline traceability.*

3. *Problem reporting: The content and identification of Problem Reports for the software product and software life cycle processes, when they will be written, the method of closing Problem Reports, and the relationship to the change control activity.*

4. *Change control: Configuration items and baselines to be controlled, when they will be controlled, the problem/change control activities that control them, pre-certification controls, post-certification controls, and the means of preserving the integrity of baselines and configuration items.*

5. *Change review: The method of handling feedback from and to the software life cycle processes; the methods of assessing and prioritizing problems, approving changes, and handling their resolution or change implementation; and the relationship of these methods to the problem reporting and change control activities.*

6. *Configuration status accounting: The data to be recorded to enable reporting configuration management status, definition of where that data will be kept, how it will be retrieved for reporting, and when it will be available.*

7. *Archive, retrieval, and release: The integrity controls, the release method and authority, and data retention.*

8. *Software load control: A description of the software load control safeguards and records.*

9. *Software life cycle environment controls: Controls for the tools used to develop, build, verify, and load the software, addressing* sections MB.11.4.b.1 through MB.11.4.b.7. *This includes control of tools to be qualified.*

10. *Software life cycle data controls: Controls associated with CC1 and CC2 data.*

c. *Transition criteria: The transition criteria for entering the SCM process.*

d. *SCM data: A definition of the software life cycle data produced by the SCM process, including SCM Records, the Software Configuration Index, and the Software Life Cycle Environment Configuration Index.*

e. *Supplier control: The means of applying SCM process requirements to suppliers.*

MB.11.5 Software Quality Assurance Plan

Section 11.5 of DO-178C is unchanged.

MB.11.6 Software Requirements Standards

Section 11.6 of DO-178C is unchanged.

Note: If high-level requirements are expressed by a model, additional standards will be addressed in the Software Model Standards (see MB.11.23).

MB.11.7 Software Design Standards

Section 11.7 of DO-178C is unchanged.

Note: If low-level requirements and/or software architecture are expressed by a model, additional standards will be addressed in the Software Model Standards (see MB.11.23).

MB.11.8 Software Code Standards

Section 11.8 of DO-178C is unchanged.

MB.11.9 Software Requirements Data

Software Requirements Data is a definition of the high-level requirements including the derived requirements. A Specification Model may partially or fully express high-level requirements. *This data should include:*

a. *Description of the allocation of system requirements to software, with attention to safety-related requirements and potential failure conditions.*

b. *Functional and operational requirements under each mode of operation.*

c. *Performance criteria, for example, precision and accuracy.*

d. *Timing requirements and constraints.*

e. *Memory size constraints.*

f. *Hardware and software interfaces, for example, protocols, formats, frequency of inputs, and frequency of outputs.*

g. *Failure detection and safety monitoring requirements.*

h. *Partitioning requirements allocated to software, how the partitioned software components interact with each other, and the software level(s) of each partition.*

Within this supplement, the term "high-level requirements" refers to either any requirement contained in a Specification Model or any requirement from which a Design Model is developed.

MB.11.10 Design Description

The Design Description is a definition of the software architecture and the low-level requirements that will satisfy the high-level requirements. A Design Model may partially or fully express software architecture and/or low-level requirements. *This data should include:*

a. A detailed description of how the software satisfies the specified high-level requirements, including algorithms, data structures, and how software requirements are allocated to processors and tasks.

b. The description of the software architecture defining the software structure to implement the requirements.

c. The input/output description, for example, a data dictionary, both internally and externally throughout the software architecture.

d. The data flow and control flow of the design.

e. Resource limitations, the strategy for managing each resource and its limitations, the margins, and the method for measuring those margins, for example, timing and memory.

f. Scheduling procedures and inter-processor/inter-task communication mechanisms, including time-rigid sequencing, preemptive scheduling, Ada rendezvous, and interrupts.

g. Design methods and details for their implementation, for example, software loading, user-modifiable software, or multiple-version dissimilar software.

h. Partitioning methods and means of preventing partition breaches.

i. Descriptions of the software components, whether they are new or previously developed, and, if previously developed, reference to the baseline from which they were taken.

j. Derived requirements resulting from the software design process.

k. If the system contains deactivated code, a description of the means to ensure that the code cannot be enabled in the target computer.

l. Rationale for those design decisions that are traceable to safety-related system requirements.

Within this supplement, the terms "software architecture" and/or "low-level requirements" can be used synonymously with Design Model for the software architecture and/or those specific low-level requirements that are expressed by the Design Model.

MB.11.11 Source Code

Section 11.11 of DO-178C is unchanged.

MB.11.12 Executable Object Code

Section 11.12 of DO-178C is unchanged.

MB.11.13 Software Verification Cases and Procedures

Software Verification Cases and Procedures detail how the software verification process activities are implemented. This data should include descriptions of the:

a. *Review and analysis procedures: The scope and depth of the review or analysis methods to be used, in addition to the description in the Software Verification Plan.*

b. *Test cases: The purpose of each test case, set of inputs, conditions, expected results to achieve the required* test *coverage criteria, and the pass/fail criteria.*

c. *Test procedures: The step-by-step instructions for how each test case is to be set up and executed, how the test results are evaluated, and the test environment to be used.*

d. Simulation cases: The purpose of each simulation case, set of inputs, conditions, expected results, and pass/fail criteria.

e. Simulation procedures: The step-by-step instructions for how each simulation case is to be set up and executed, how the simulation results are evaluated, and the simulation environment to be used.

MB.11.14 Software Verification Results

The Software Verification Results are produced by the software verification process activities. Software Verification Results should:

a. *For each review, analysis,* simulation, *and test, indicate each procedure that passed or failed during the activities and the final pass/fail results.*

b. *Identify the configuration item,* model, *or software version reviewed, analyzed,* simulated, *or tested.* If utilizing a model representing high-level requirements, low-level requirements, and/or software architecture, the configuration of the model should be identified.

c. *Include the results of tests,* simulations, *reviews, and analyses, including coverage analyses and traceability analyses*, and any analysis results in support of simulation.

Any discrepancies found should be recorded and tracked via problem reporting.

Additionally, evidence provided in support of the system processes' assessment of the information provided by the software process (see MB.2.2.1.f *and* MB.2.2.1.g*) should be considered to be Software Verification Results.*

MB.11.15 Software Life Cycle Environment Configuration Index

The Software Life Cycle Environment Configuration Index (SECI) identifies the configuration of the software life cycle environment. This index is written to aid reproduction of the hardware and software life cycle environment, for software regeneration, reverification, or software modification, and should:

a. *Identify the software life cycle environment hardware and its operating system software.*

b. *Identify the tools to be used during the development of the software. Examples include compilers, linkage editors, loaders, data integrity tools such as tools that*

calculate and embed checksums or cyclical redundancy checks, and any autocode generator with its associated options.

c. *Identify the test* and simulation *environments,* and associated settings*, used to verify the software product, for example, the software testing and analysis tools.*

d. *Identify qualified tools and their associated tool qualification data.*

Note: This data may be included in the Software Configuration Index.

MB.11.16 Software Configuration Index

The Software Configuration Index (SCI) identifies the configuration of the software product. Specific configuration identifiers and version identifiers should be provided.

Note: The SCI can contain one data item or a set (hierarchy) of data items. The SCI can contain the items listed below or it may reference another SCI or other configuration identified data that specifies the individual items and their versions.

The SCI should identify:

a. *The software product.*

b. *Executable Object Code and Parameter Data Item Files, if any.*

c. *Each Source Code component.*

d. *Previously developed software in the software product, if used.*

e. *Software life cycle data.* This data should address section items MB.2.2.1.i through MB.2.2.1.o. If a system configuration index exists and includes that data, a reference to this system configuration index is considered sufficient.

f. *Archive and release media.*

g. *Instructions for building the Executable Object Code and Parameter Data Item Files, if any, including, for example, instructions and data for compiling and linking; and the procedures used to recover the software for regeneration, testing, or modification.*

h. *Reference to the Software Life Cycle Environment Configuration Index (see* MB.11.15*), if it is packaged separately.*

i. *Data integrity checks for the Executable Object Code, if used.*

j. *Procedures, methods, and tools for making modifications to the user-modifiable software, if any.*

k. *Procedures and methods for loading the software into the target hardware.*

Note: The SCI may be produced for one software product version or it may be extended to contain data for several alternative or successive software product versions.

If utilizing a model representing high-level requirements, low-level requirements, and/or software architecture, the configuration of the model should be identified.

MB.11.17 Problem Reports

Section 11.17 of DO-178C is unchanged.

MB.11.18 Software Configuration Management Records

The results of the SCM process activities are recorded in SCM Records. Examples include configuration identification lists, baseline or software library records, change history reports, archive records, and release records. Examples of SCM Records for models include baseline records and model element library records. *These examples do not imply that records of these specific types need to be produced.*

Note: Due to the integral nature of the SCM process, its outputs will often be included as parts of other software life cycle data.

MB.11.19 Software Quality Assurance Records

Section 11.19 of DO-178C is unchanged.

MB.11.20 Software Accomplishment Summary

Section 11.20 of DO-178C is unchanged.

MB.11.21 Trace Data

Trace Data establishes the associations between life cycle data items contents. Trace Data should be provided that demonstrates bi-directional associations between:

a. System requirements allocated to software and high-level requirements.

b. High-level requirements and low-level requirements.

c. Low-level requirements and Source Code.

d. Software Requirements and test cases.

e. Test cases and test procedures.

f. Test procedures and test results.

If utilizing a model representing high-level requirements or low-level requirements, and/or software architecture, traceability with the model should provide the granularity to demonstrate the above.

MB.11.22 Parameter Data Item File

Section 11.22 of DO-178C is unchanged.

MB.11.23 Software Model Standards

Software Model Standards define modeling techniques for each type of model. The definition of each modeling technique should include:

a. Methods and tools used for developing the models.

b. Modeling languages to be used. For each modeling language, reference the data that unambiguously defines the syntax and semantics of the language with respect to the type of software life cycle data the model represents. This may require limiting the use of some features of the modeling language.

c. Style guidelines and complexity restrictions for the use of the modeling languages and tools, for example, naming conventions, diagram layout, allowable elements, maximum number of nesting levels, maximum number of model elements per diagram, and maximum number of architectural layers.

d. Constraints on the use of the modeling tool(s) (for example, scheduling methods, and/or global data) and use of model element libraries.

e. Method to be used to identify and delimit the requirements contained in the model and the means to establish traceability between requirements and other life cycle data.

f. Method to be used to identify and delimit the derived requirements contained in the model and the method to provide derived requirements to the system processes, including the system safety assessment process.

g. Means to identify each model element that does not contribute to the representation of a software requirement or of the software architecture and is not an input to a subsequent software development process or activity, for example, a comment block.

h. Rationale for the suitability of the technique for the type of information expressed by a Specification Model or Design Model.

This Page Intentionally Left Blank

MB.12.0 ADDITIONAL CONSIDERATIONS

Section 12.0 of DO-178C is unchanged.

MB.12.1 Use of Previously Developed Software

Section 12.1 of DO-178C is unchanged.

MB.12.1.1 Modifications of Previously Developed Software

This guidance discusses modifications to previously developed software where the outputs of the previous software life cycle processes comply with this document. Modification may result from requirement changes, the detection of errors, and/or software enhancements.

Activities include:

a. *The revised outputs of the system safety assessment process should be reviewed considering the proposed modifications.*

b. *If the software level is revised, the guidance of section* MB.12.1.4 *should be considered.*

c. *Both the impact of the software requirements changes and the impact of software architecture changes should be analyzed, including the consequences of software requirement changes upon other requirements and the coupling between several software components that may result in reverification effort involving more than the modified area.*

d. *The area affected by a change should be determined. This may be done by data flow analysis, control flow analysis, timing analysis, traceability analysis, or a combination of these analyses.*

e. *Areas affected by the change should be reverified in accordance with section* MB.6.

MB.12.1.2 Change of Aircraft Installation

Airborne systems or equipment containing software that has been previously "approved" at a certain software level and under a specific certification basis may be used in a new aircraft installation. Activities include:

a. *The system safety assessment process assesses the new aircraft installation and determines the software level and the certification basis. No additional effort will be required if these are the same for the new installation as they were in the previous installation.*

b. *If functional modifications are required for the new installation, the guidance of section* MB.12.1.1 *should be satisfied.*

c. *If the previous development activity did not produce outputs required to substantiate the system safety objectives of the new installation, the guidance of section* MB.12.1.4 *should be satisfied.*

MB.12.1.3 Change of Application or Development Environment

Use and modification of previously developed software may involve a new development environment, a new target processor or other hardware, or integration with other software than that used for the original application.

New development environments may increase or reduce some activities within the software life cycle. New application environments may require activities in addition to software life cycle process activities that address modifications.

Changes to an application or development environment should be identified, analyzed, and reverified.

Activities include:

a. *If a new development environment uses software tools, the guidance of section* MB.12.2 *may be applicable.*

b. *The rigor of the evaluation of an application change should consider the complexity and sophistication of the programming language. For example, the rigor of the evaluation for Ada generics will be greater if the generic parameters are different in the new application. For object-oriented languages, the rigor will be greater if the objects that are inherited are different in the new application.*

c. *Using a different autocode generator or a different set of autocode generator options may change the Source Code or object code generated. The impact of any changes should be analyzed.*

d. *If a different compiler or different set of compiler options are used, resulting in different object code, the results from a previous software verification process activity using the object code may not be valid and should not be used for the new application. In this case, previous test results may no longer be valid for the structural coverage criteria of the new application. Similarly, compiler assumptions about optimization may not be valid.*

e. *If a different processor is used, then a change impact analysis is performed to determine:*

    1. *Software components that are new or will need to be modified as a result of changing the processor, including any modification for hardware/software integration.*

    2. *The results from a previous software verification process activity directed at the hardware/software interface that may be used for the new application.*

    3. *Previous hardware/software integration tests that should be executed for the new application. It is expected that there will always be a minimal set of tests to be run.*

    4. *Additional hardware/software integration tests and reviews that may be necessary.*

f. *If a hardware item, other than the processor, is changed and the design of the software isolates the interfacing modules from other modules then a change impact analysis should be performed to:*

1. *Determine the software modules or interfaces that are new or will be modified to accommodate the changed hardware component.*

2. *Determine the extent of reverification required.*

g. *Verification of software interfaces should be conducted where previously developed software is used with different interfacing software. A change impact analysis may be used to determine the extent of reverification required.*

MB.12.1.4 Upgrading a Development Baseline

Guidance follows for software whose software life cycle data from a previous application are determined to be inadequate or do not satisfy the objectives of this document, due to the requirements associated with a new application. This guidance is intended to aid in satisfying the objectives of this document when applied to:

- *COTS software.*
- *Airborne software developed to other guidance.*
- *Airborne software developed prior to the existence of this document.*
- *Software previously developed to this document at a lower software level.*

Activities for upgrading a development baseline include:

a. *The objectives of this document should be satisfied while taking advantage of software life cycle data of the previous development that satisfy the objectives for the new application.*

b. *Software aspects of certification should be based on the failure conditions and software level(s) as determined by the system safety assessment process. Comparison to failure conditions of the previous application will determine areas that may need to be upgraded.*

c. *Software life cycle data from a previous development should be evaluated to ensure that the software verification process objectives of the software level are satisfied for the new application to the necessary level of rigor and independence.*

d. *Reverse engineering may be used to regenerate software life cycle data that is inadequate or missing in satisfying the objectives of this document. In addition to producing the software product, additional activities may need to be performed to satisfy the software verification process objectives.*

e. *If use of product service history is planned to satisfy the objectives of this document in upgrading a development baseline, section* MB.12.3.4 *should be considered.*

f. *The applicant should specify the strategy for accomplishing compliance with this document in the Plan for Software Aspects of Certification.*

MB.12.1.5 Software Configuration Management Considerations

If previously developed software is used, the software configuration management process activities for the new application should include, in addition to the activities of section MB.7:

a. *Providing traceability from the software product and software life cycle data of the previous application to the new application.*

b. *Providing change control that enables problem reporting, problem resolution, and tracking of changes to software components used in more than one application.*

MB.12.1.6 Software Quality Assurance Considerations

If previously developed software is used, the software quality assurance activities should include, in addition to the activities of section MB.8*:*

a. *Providing assurance that the software components satisfy or exceed the software life cycle criteria of the software level for the new application.*

b. *Providing assurance that changes to the software life cycle processes are stated in the software plans.*

MB.12.2 Tool Qualification

MB.12.2.1 Determining if Tool Qualification is Needed

Qualification of a tool is needed when processes of this document are eliminated, reduced, or automated by the use of a software tool without its output being verified as specified in section MB.6.

The purpose of the tool qualification process is to ensure that the tool provides confidence at least equivalent to that of the process(es) eliminated, reduced, or automated.

The tool qualification process may be applied to a single tool, a collection of tools, or one or more functions within a tool. For a tool with multiple functions, if protection of tool functions can be demonstrated, only those functions that are used to eliminate, reduce or automate software life cycle processes, and whose outputs are not verified, need be qualified. Protection is the use of a mechanism to ensure that a tool function cannot adversely impact another tool function.

A tool is qualified only for use on a specific system where the intention to use the tool is stated in the Plan for Software Aspects of Certification that supports the system. If a tool previously qualified on one system is proposed for use on another system, it should be re-qualified within the context of that other system.

MB.12.2.2 Determining the Tool Qualification Level

Section 12.2.2 of DO-178C is unchanged.

MB.12.2.3 Tool Qualification Process

Section 12.2.3 of DO-178C is unchanged.

MB.12.3 Alternative Methods

Section 12.3 of DO-178C is unchanged.

MB.12.3.1 Exhaustive Input Testing

There are situations where the software component of an airborne system or equipment is simple and isolated such that the set of inputs and outputs can be bounded. If so, it may be possible to demonstrate that exhaustive testing of this input space can be substituted for one or more of the software verification process activities identified in section MB.6.

For this alternative method, activities include:

a. Defining the complete set of valid inputs and outputs of the software.

b. Performing an analysis that confirms the isolation of the inputs to the software.

c. Developing rationale for the exhaustive input test cases and procedures.

d. Developing the test cases, test procedures, and test results.

MB.12.3.2 Considerations for Multiple-Version Dissimilar Software Verification

Section 12.3.2 of DO-178C is unchanged.

MB.12.3.3 Software Reliability Models

Section 12.3.3 of DO-178C is unchanged.

MB.12.3.4 Product Service History

Section 12.3.4 of DO-178C is unchanged.

MB.12.3.4.1 Relevance of Service History

Section 12.3.4.1 of DO-178C is unchanged.

MB.12.3.4.2 Sufficiency of Accumulated Service History

The required amount of service history is determined by:

a. The system safety objectives of the software and the software level.

b. Any differences in service history environment and system operational environment.

c. The objectives from sections MB.4 to MB.9 *being addressed by service history.*

d. Evidence, in addition to service history, addressing those objectives.

MB.12.3.4.3 Collection, Reporting, and Analysis of Problem Reports Found During Service History

Section 12.3.4.3 of DO-178C is unchanged.

MB.12.3.4.4 Service History Information to be Included in the Plan for Software Aspects of Certification

The following items should be specified in the Plan for Software Aspects of Certification and agreed with the certification authority when seeking certification credit for service history:

a. *Rationale for claiming relevant service history, addressing the items in* DO-178C *section 12.3.4.1.*

b. *Amount of service history needed together with the rationale. This should include the items in section* MB.12.3.4.2*, any censoring rules for data used in estimation, and measured parameters, if applicable. This data should be provided using measures relevant to the operations of the system.*

c. *Rationale for calculating the total relevant service history period, including factors such as operational modes, the number of independently operating copies in the installation and in service, and the definition of "normal operation" and "normal operation time."*

   Note: If the error rate is greater than that identified in the plan, these errors should be analyzed and the analyses reviewed with the certification authority. The length of the service history period may need to be extended or service history may be inapplicable as an alternative means of compliance.

d. *Definition of what was counted as an error and rationale for that definition. This should address the items in* DO-178C *section 12.3.4.3.a.*

e. *Proposed acceptable error rates and rationale for the product service history period in relation to the system safety and proposed error rates. This should address the items in* DO-178C *sections 12.3.4.3.b and 12.3.4.3.c.*

f. *Definition of criteria for problems that would invalidate service history under* DO-178C *section 12.3.4.3 or for other reasons.*

g. *Criteria for errors that will be corrected; how they will be corrected and verified; and rationale for any defects for which no action will be taken.*

h. *Objectives in sections* MB.4 to MB.9 *to be addressed through the use of service history.*

ANNEX MB.A – PROCESS OBJECTIVES AND OUTPUTS BY SOFTWARE LEVEL IN DO-178C

This annex provides tables that describe the objectives, related activities, and outputs given in DO-178C Annex A that are relevant when using model-based development and verification in the production of airborne software.

The tables include guidance for:

a. *The process objectives applicable for each software level. For level E software, see* DO-178C *section 2.3.3.*

b. *The independence by software level of the software life cycle process activities applicable to satisfy that process's objectives.*

c. *The control category by software level for the software life cycle data produced by the software life cycle process activities (section* MB.7.3*).*

The tables include references as appropriate to DO-178C and to this supplement. For clear identification, references to DO-178C do not include a prefix, whereas references to this supplement include "MB." as a prefix.

These tables should not be used as a checklist. These tables do not reflect all aspects of compliance to this document. In order to fully understand the guidance, the full body of DO-178C and this document *should be considered.*

The following legend applies to "Applicability by Software Level" and "Control Category by Software Level" for all tables:

LEGEND:	●	*The objective should be satisfied with independence.*
	○	*The objective should be satisfied.*
	Blank	*Satisfaction of objective is at applicant's discretion.*
	①	*Data satisfies the objectives of Control Category 1 (CC1).*
	②	*Data satisfies the objectives of Control Category 2 (CC2).*

Table MB.A-1 Software Planning Process

	Objective		Activity	Applicability by Software Level				Output		Control Category by Software Level			
	Description	Ref	Ref	A	B	C	D	Data Item	Ref	A	B	C	D
1	The activities of the software life cycle processes are defined.	MB.4.1.a	MB.4.2.a MB.4.2.c MB.4.2.d MB.4.2.e MB.4.2.g MB.4.2.i MB.4.2.l MB.4.2.m MB.4.2.n MB.4.2.o MB.4.3.c	❍	❍	❍	❍	*PSAC* *SDP* *SVP* *SCM Plan* *SQA Plan*	MB.11.1 MB.11.2 MB.11.3 MB.11.4 *11.5*	① ① ① ① ① ① ② ② ① ① ② ② ① ① ② ② ① ① ② ②			
2	*The software life cycle(s), including the inter-relationships between the processes, their sequencing, feedback mechanisms, and transition criteria, is defined.*	MB.4.1.b	MB.4.2.i MB.4.3.b	❍	❍	❍		*PSAC* *SDP* *SVP* *SCM Plan* *SQA Plan*	MB.11.1 MB.11.2 MB.11.3 MB.11.4 *11.5*	① ① ① ① ① ② ① ① ② ① ① ② ① ① ②			
3	*Software life cycle environment is selected and defined.*	MB.4.1.c	*4.4.1* MB.4.4.2.a MB.4.4.2.b MB.4.4.2.c MB.4.4.3 MB.4.4.4	❍	❍	❍		*PSAC* *SDP* *SVP* *SCM Plan* *SQA Plan*	MB.11.1 MB.11.2 MB.11.3 MB.11.4 *11.5*	① ① ① ① ① ② ① ① ② ① ① ② ① ① ②			
4	*Additional considerations are addressed.*	MB.4.1.d	MB.4.2.f MB.4.2.h MB.4.2.i MB.4.2.j MB.4.2.k	❍	❍	❍	❍	*PSAC* *SDP* *SVP* *SCM Plan* *SQA Plan*	MB.11.1 MB.11.2 MB.11.3 MB.11.4 *11.5*	① ① ① ① ① ① ② ② ① ① ② ② ① ① ② ② ① ① ② ②			
5	*Software development standards are defined.*	MB.4.1.e	MB.4.2.b MB.4.2.g MB.4.5	❍	❍	❍		*SW Requirements Standards* *SW Design Standards* *SW Code Standards* SW Model Standards	*11.6* *11.7* *11.8* MB.11.23	① ① ② ① ① ② ① ① ② ① ① ②			
6	*Software plans comply with this document.*	MB.4.1.f	MB.4.3.a MB.4.6	❍	❍	❍		*Software Verification Results*	MB.11.14	② ② ②			
7	*Development and revision of software plans are coordinated.*	MB.4.1.g	MB.4.2.g MB.4.6	❍	❍	❍		*Software Verification Results*	MB.11.14	② ② ②			

Table MB.A-2 Software Development Processes

	Objective		Activity	Applicability by Software Level				Output		Control Category by Software Level			
	Description	Ref	Ref	A	B	C	D	Data Item	Ref	A	B	C	D
1	*High-level requirements are developed.*	MB.5.1.1.a	MB.5.1.2.a MB.5.1.2.b MB.5.1.2.c MB.5.1.2.d MB.5.1.2.e MB.5.1.2.f MB.5.1.2.g MB.5.1.2.j MB.5.1.2.k MB.5.1.2.l MB.5.5 *5.5.a*	❍	❍	❍	❍	*Software Requirements Data* *Trace Data*	MB.11.9 MB.11.21	① ①	① ①	① ①	① ①
2	*Derived high-level requirements are defined and provided to the system processes, including the system safety assessment process.*	MB.5.1.1.b	MB.5.1.2.h MB.5.1.2.i MB.5.1.2.k	❍	❍	❍	❍	*Software Requirements Data*	MB.11.9	①	①	①	①
3	*Software architecture is developed.*	MB.5.2.1.a	MB.5.2.2.a MB.5.2.2.d MB.5.2.2.h	❍	❍	❍	❍	*Design Description*	MB.11.10	①	①	①	②
4	*Low-level requirements are developed.*	MB.5.2.1.a	MB.5.2.2.a MB.5.2.2.e MB.5.2.2.f MB.5.2.2.g MB.5.2.2.h MB.5.2.3.a MB.5.2.3.b *5.2.4.a* *5.2.4.b* *5.2.4.c* MB.5.5 *5.5.b*	❍	❍	❍		*Design Description* *Trace Data*	MB.11.10 MB.11.21	① ①	① ①	① ①	
5	*Derived low-level requirements are defined and provided to the system processes, including the system safety assessment process.*	MB.5.2.1.b	MB.5.2.2.b MB.5.2.2.c MB.5.2.2.h	❍	❍	❍		*Design Description*	MB.11.10	①	①	①	
6	*Source Code is developed.*	*5.3.1.a*	*5.3.2.a* *5.3.2.b* *5.3.2.c* *5.3.2.d* MB.5.5 *5.5.c*	❍	❍	❍		*Source Code* *Trace Data*	*11.11* MB.11.21	① ①	① ①	① ①	

Table MB.A-2 (Continued) Software Development Processes

	Objective		Activity	Applicability by Software Level				Output		Control Category by Software Level			
	Description	Ref	Ref	A	B	C	D	Data Item	Ref	A	B	C	D
7	*Executable Object Code and Parameter Data Item Files, if any, are produced and loaded in the target computer.*	*5.4.1.a*	*5.4.2.a 5.4.2.b 5.4.2.c 5.4.2.d 5.4.2.e 5.4.2.f*	❍	❍	❍	❍	*Executable Object Code* *Parameter Data Item File*	*11.12* *11.22*	① ①	① ①	① ①	① ①
MB 8	Specification Model elements that do not contribute to implementation or realization of any high-level requirement are identified.	MB.5.1.1.c	MB.5.1.2.k	❍	❍	❍	❍	Software Requirements Data	MB.11.9	①	①	①	①
MB 9	Design Model elements that do not contribute to implementation or realization of any software architecture are identified.	MB.5.2.1.c	MB.5.2.2.h	❍	❍	❍	❍	Design Description	MB.11.10	①	①	①	②
MB 10	Design Model elements that do not contribute to implementation or realization of any low-level requirement are identified.	MB.5.2.1.c	MB.5.2.2.h	❍	❍	❍		Design Description	MB.11.10	①	①	①	

Table MB.A-3 Verification of Outputs of Software Requirements Process

	Objective		Activity	Applicability by Software Level				Output		Control Category by Software Level			
	Description	Ref	Ref	A	B	C	D	Data Item	Ref	A	B	C	D
1	*High-level requirements comply with system requirements.* (See Item 1)	MB.6.3.1.a	MB.6.3.1 MB.6.8.1 (See Item 2)	●	●	❍	❍	*Software Verification Results*	MB.11.14	②	②	②	②
2	*High-level requirements are accurate and consistent.*	MB.6.3.1.b	MB.6.3.1 MB.6.8.1 (See Item 2)	●	●	❍	❍	*Software Verification Results*	MB.11.14	②	②	②	②
3	*High-level requirements are compatible with target computer.*	MB.6.3.1.c	MB.6.3.1	❍	❍			*Software Verification Results*	MB.11.14	②	②		
4	*High-level requirements are verifiable.*	MB.6.3.1.d	MB.6.3.1 MB.6.8.1 (See Item 2)	❍	❍	❍		*Software Verification Results*	MB.11.14	②	②	②	
5	*High-level requirements conform to standards.*	MB.6.3.1.e	MB.6.3.1	❍	❍	❍		*Software Verification Results*	MB.11.14	②	②	②	
6	*High-level requirements are traceable to system requirements.* (See Item 1)	MB.6.3.1.f	MB.6.3.1	❍	❍	❍	❍	*Software Verification Results*	MB.11.14	②	②	②	②
7	*Algorithms are accurate.*	MB.6.3.1.g	MB.6.3.1 MB.6.8.1 (See Item 2)	●	●	❍		*Software Verification Results*	MB.11.14	②	②	②	
MB 8	Simulation cases are correct. (See Item 2)	MB.6.8.3.2.a	MB.6.8.1 MB.6.8.3.2	●	❍	❍	❍	Software Verification Results	MB.11.14	②	②	②	②
MB 9	Simulation procedures are correct. (See Item 2)	MB.6.8.3.2.b	MB.6.8.1 MB.6.8.3.2	●	❍	❍	❍	Software Verification Results	MB.11.14	②	②	②	②
MB 10	Simulation results are correct and discrepancies explained. (See Item 2)	MB.6.8.3.2.c	MB.6.8.1 MB.6.8.3.2	●	❍	❍	❍	Software Verification Results	MB.11.14	②	②	②	②

Item 1: In case the requirements from which a Design Model is developed are output of the system process, the objectives 1 and 6 are implicitly satisfied for these requirements.

Item 2: As described in section MB.6.8.1 of the supplement, simulation may be used as a means of compliance for objectives 1, 2, 4 or 7 of this table. If simulation is used as this means, objectives MB.8, MB.9, and MB.10 are required.

Table MB.A-4 Verification of Outputs of Software Design Process

	Objective		Activity	Applicability by Software Level				Output		Control Category by Software Level			
	Description	Ref	Ref	A	B	C	D	Data Item	Ref	A	B	C	D
1	*Low-level requirements comply with high-level requirements.*	MB.6.3.2.a	MB.6.3.2 MB.6.7 MB.6.8.1 (See Item 1)	●	●	❍		*Software Verification Results*	MB.11.14	②	②	②	
2	*Low-level requirements are accurate and consistent.*	MB.6.3.2.b	MB.6.3.2 MB.6.8.1 (See Item 1)	●	●	❍		*Software Verification Results*	MB.11.14	②	②	②	
3	*Low-level requirements are compatible with target computer.*	MB.6.3.2.c	MB.6.3.2	❍	❍			*Software Verification Results*	MB.11.14	②	②		
4	*Low-level requirements are verifiable.*	MB.6.3.2.d	MB.6.3.2 MB.6.8.1 (See Item 1)	❍	❍			*Software Verification Results*	MB.11.14	②	②		
5	*Low-level requirements conform to standards.*	MB.6.3.2.e	MB.6.3.2	❍	❍	❍		*Software Verification Results*	MB.11.14	②	②	②	
6	*Low-level requirements are traceable to high-level requirements.*	MB.6.3.2.f	MB.6.3.2	❍	❍	❍		*Software Verification Results*	MB.11.14	②	②	②	
7	*Algorithms are accurate.*	MB.6.3.2.g	MB.6.3.2 MB.6.8.1 (See Item 1)	●	●	❍		*Software Verification Results*	MB.11.14	②	②	②	
8	*Software architecture is compatible with high-level requirements.*	MB.6.3.3.a	MB.6.3.3 MB.6.8.1 (See Item 1)	●	❍	❍		*Software Verification Results*	MB.11.14	②	②	②	
9	*Software architecture is consistent.*	MB.6.3.3.b	MB.6.3.3 MB.6.8.1 (See Item 1)	●	❍	❍		*Software Verification Results*	MB.11.14	②	②	②	
10	*Software architecture is compatible with target computer.*	MB.6.3.3.c	MB.6.3.3	❍	❍			*Software Verification Results*	MB.11.14	②	②		
11	*Software architecture is verifiable.*	MB.6.3.3.d	MB.6.3.3 MB.6.8.1 (See Item 1)	❍	❍			*Software Verification Results*	MB.11.14	②	②		
12	*Software architecture conforms to standards.*	MB.6.3.3.e	MB.6.3.3	❍	❍	❍		*Software Verification Results*	MB.11.14	②	②	②	
13	*Software partitioning integrity is confirmed.*	MB.6.3.3.f	MB.6.3.3	●	❍	❍	❍	*Software Verification Results*	MB.11.14	②	②	②	②
MB 14	Simulation cases are correct. (See Item 1)	MB.6.8.3.2.a	MB.6.8.1 MB.6.8.3.2	●	❍	❍		Software Verification Results	MB.11.14	②	②	②	

Table MB.A-4 (Continued) Verification of Outputs of Software Design Process

	Objective		Activity	Applicability by Software Level				Output		Control Category by Software Level			
	Description	Ref	Ref	A	B	C	D	Data Item	Ref	A	B	C	D
MB 15	Simulation procedures are correct. (See Item 1)	MB.6.8.3.2.b	MB.6.8.1 MB.6.8.3.2	●	❍	❍		Software Verification Results	MB.11.14	②	②	②	
MB 16	Simulation results are correct and discrepancies explained. (See Item 1)	MB.6.8.3.2.c	MB.6.8.1 MB.6.8.3.2	●	❍	❍		Software Verification Results	MB.11.14	②	②	②	

Item 1: As described in section MB.6.8.1 of this supplement, simulation may be used as a means of compliance for objectives 1, 2, 4, 7, 8, 9, or 11 of this table. If simulation is used as this means, objectives MB.14, MB.15 and MB.16 are required.

Table MB.A-5 Verification of Outputs of Software Coding & Integration Processes

	Objective		Activity	Applicability by Software Level				Output		Control Category by Software Level			
	Description	Ref	Ref	A	B	C	D	Data Item	Ref	A	B	C	D
1	*Source Code complies with low-level requirements.*	MB.6.3.4.a	MB.6.3.4	●	●	❍		*Software Verification Results*	MB.11.14	②	②	②	
2	*Source Code complies with software architecture.*	MB.6.3.4.b	MB.6.3.4	●	❍	❍		*Software Verification Results*	MB.11.14	②	②	②	
3	*Source Code is verifiable.*	MB.6.3.4.c	MB.6.3.4	❍	❍			*Software Verification Results*	MB.11.14	②	②		
4	*Source Code conforms to standards.*	MB.6.3.4.d	MB.6.3.4	❍	❍	❍		*Software Verification Results*	MB.11.14	②	②	②	
5	*Source Code is traceable to low-level requirements.*	MB.6.3.4.e	MB.6.3.4	❍	❍	❍		*Software Verification Results*	MB.11.14	②	②	②	
6	*Source Code is accurate and consistent.*	MB.6.3.4.f	MB.6.3.4	●	❍	❍		*Software Verification Results*	MB.11.14	②	②	②	
7	*Output of software integration process is complete and correct.*	*6.3.5 a*	*6.3.5*	❍	❍	❍		*Software Verification Results*	MB.11.14	②	②	②	
8	*Parameter Data Item File is correct and complete.*	*6.6.a*	*6.6*	●	●	○	○	Software Verification Cases and Procedures *Software Verification Results*	MB.11.13 MB.11.14	① ②	① ②	② ②	② ②
9	*Verification of Parameter Data Item File is achieved.*	6.6.b	*6.6*	●	●	○		*Software Verification Results*	MB.11.14	②	②	②	

Table MB.A-6 Testing of Outputs of Integration Process

	Objective		Activity	Applicability by Software Level				Output		Control Category by Software Level			
	Description	Ref	Ref	A	B	C	D	Data Item	Ref	A	B	C	D
1	*Executable Object Code complies with high-level requirements.*	*6.4.a*	*6.4.2* *6.4.2.1* *6.4.3* *6.5* MB.6.8.2.a (See Item 1)	❍	❍	❍	❍	*Software Verification Cases and Procedures* *Software Verification Results* *Trace Data*	MB.11.13 MB.11.14 MB.11.21	① ② ①	① ② ①	② ② ②	② ② ②
2	*Executable Object Code is robust with high-level requirements.*	*6.4.b*	*6.4.2* *6.4.2.2* *6.4.3* *6.5* MB.6.8.2.a (See Item 1)	❍	❍	❍	❍	*Software Verification Cases and Procedures* *Software Verification Results* *Trace Data*	MB.11.13 MB.11.14 MB.11.21	① ② ①	① ② ①	② ② ②	② ② ②
3	*Executable Object Code complies with low-level requirements.*	*6.4.c*	*6.4.2* *6.4.2.1* *6.4.3* *6.5*	●	●	❍		*Software Verification Cases and Procedures* *Software Verification Results* *Trace Data*	MB.11.13 MB.11.14 MB.11.21	① ② ①	① ② ①	② ② ②	
4	*Executable Object Code is robust with low-level requirements.*	*6.4.d*	*6.4.2* *6.4.2.2* *6.4.3* *6.5*	●	❍	❍		*Software Verification Cases and Procedures* *Software Verification Results* *Trace Data*	MB.11.13 MB.11.14 MB.11.21	① ② ①	① ② ①	② ② ②	
5	*Executable Object Code is compatible with target computer.*	*6.4.e*	*6.4.1 a* *6.4.3.a*	❍	❍	❍	❍	*Software Verification Cases and Procedures* *Software Verification Results*	MB.11.13 MB.11.14	① ②	① ②	② ②	② ②

Item 1: As described in section MB.6.8.2.a of the supplement, the MB.6.8.2.a activity is only required when simulation is used as a means of compliance for objectives 1 or 2 of this table.

Table MB.A-7 Verification of Verification Process Results

	Objective		Activity	Applicability by Software Level				Output		Control Category by Software Level			
	Description	Ref	Ref	A	B	C	D	Data Item	Ref	A	B	C	D
1	*Test procedures are correct.*	*6.4.5.b*	*6.4.5*	●	❍	❍		*Software Verification Results*	MB.11.14	②	②	②	
2	*Test results are correct and discrepancies explained.*	*6.4.5.c*	*6.4.5*	●	❍	❍		*Software Verification Results*	MB.11.14	②	②	②	
3	*Test coverage of high-level requirements is achieved.*	*6.4.4.a*	*6.4.4.1* MB.6.8.2.a	●	❍	❍	❍	*Software Verification Results*	MB.11.14	②	②	②	②
4	*Test coverage of low-level requirements is achieved.*	*6.4.4.b*	*6.4.4.1* MB.6.7	●	❍	❍		*Software Verification Results*	MB.11.14	②	②	②	
5	*Test coverage of software structure (modified condition/decision) is achieved.*	*6.4.4.c*	*6.4.4.2.a* *6.4.4.2.b* *6.4.4.2.d* *6.4.4.3* MB.6.8.2.b (See Item 1)	●				*Software Verification Results*	MB.11.14	②			
6	*Test coverage of software structure (decision coverage) is achieved.*	*6.4.4.c*	*6.4.4.2.a* *6.4.4.2.b* *6.4.4.2.d* *6.4.4.3* MB.6.8.2.b (See Item 1)	●	●			*Software Verification Results*	MB.11.14	②	②		
7	*Test coverage of software structure (statement coverage) is achieved.*	*6.4.4.c*	*6.4.4.2.a* *6.4.4.2.b* *6.4.4.2.d* *6.4.4.3* MB.6.8.2.b (See Item 1)	●	●	❍		*Software Verification Results*	MB.11.14	②	②	②	
8	*Test coverage of software structure (data coupling and control coupling) is achieved.*	*6.4.4.d*	*6.4.4.2.c* *6.4.4.2 d* *6.4.4.3* MB.6.8.2.b (See Item 1)	●	●	❍		*Software Verification Results*	MB.11.14	②	②	②	
9	*Verification of additional code, that cannot be traced to Source Code, is achieved.*	*6.4.4.c*	*6.4.4.2.b*	●				*Software Verification Results*	MB.11.14	②			
MB 10	Simulation cases are correct. (See Item 2)	MB.6.8.3.2.a	MB.6.8.3.2	●	❍	❍		Software Verification Results	MB.11.14	②	②	②	

Table MB.A-7 (Continued) Verification of Verification Process Results

	Objective		Activity	Applicability by Software Level				Output		Control Category by Software Level			
	Description	Ref	Ref	A	B	C	D	Data Item	Ref	A	B	C	D
MB 11	Simulation procedures are correct. (See Item 2)	MB.6.8.3.2.b	MB.6.8.3.2	●	❍	❍		Software Verification Results	MB.11.14	②	②	②	
MB 12	Simulation results are correct and discrepancies explained. (See Item 2)	MB.6.8.3.2.c	MB.6.8.3.2	●	❍	❍		Software Verification Results	MB.11.14	②	②	②	

Item 1: As described in section MB.6.8.2 of the supplement, the MB.6.8.2.b activity is only required when simulation is used as a means of compliance for any objectives 5, 6, 7, or 8 of this table.

Item 2: As described in section MB.6.8.2 of this supplement, these three objectives are only required when simulation is used as a means of compliance for objectives 1 or 2 of Annex Table MB.A-6.

Table MB.A-8 Software Configuration Management Process

	Objective		Activity	Applicability by Software Level				Output		Control Category by Software Level			
	Description	Ref	Ref	A	B	C	D	Data Item	Ref	A	B	C	D
1	*Configuration items are identified.*	MB.7.1.a	*7.2.1*	❍	❍	❍	❍	*SCM Records*	MB.11.18	②	②	②	②
2	*Baselines and traceability are established.*	MB.7.1.b	MB.7.2.2	❍	❍	❍	❍	*Software Configuration Index* *SCM Records*	MB.11.16 MB.11.18	① ②	① ②	① ②	① ②
3	*Problem reporting, change control, change review, and configuration accounting are established.*	MB.7.1.c MB.7.1.d MB.7.1.e MB.7.1.f	*7.2.3* *7.2.4* *7.2.5* *7.2.6*	❍	❍	❍	❍	*Problem Reports* *SCM Records*	*11.17* MB.11.18	② ②	② ②	② ②	② ②
4	*Archive, retrieval, and release are established.*	MB.7.1.g	*7.2.7*	❍	❍	❍	❍	*SCM Records*	MB.11.18	②	②	②	②
5	*Software load control is established.*	MB.7.1.h	*7.4*	❍	❍	❍	❍	*SCM Records*	MB.11.18	②	②	②	②
6	*Software life cycle environment control is established.*	MB.7.1.i	MB.7.5	❍	❍	❍	❍	*Software Life Cycle Environment Configuration Index* *SCM Records*	MB.11.15 MB.11.18	① ②	① ②	① ②	② ②

Table MB.A-9 Software Quality Assurance Process

	Objective		Activity	Applicability by Software Level				Output		Control Category by Software Level			
	Description	Ref	Ref	A	B	C	D	Data Item	Ref	A	B	C	D
1	*Assurance is obtained that software plans and standards are developed and reviewed for compliance with* DO-178C and this Supplement *and for consistency.*	MB.8.1.a	MB.8.2.b MB.8.2.h MB.8.2.i	●	●	●		*SQA Records*	*11.19*	②	②	②	
2	*Assurance is obtained that software life cycle processes comply with approved software plans.*	MB.8.1.b	MB.8.2.a MB.8.2.c MB.8.2.d MB.8.2.f MB.8.2.h MB.8.2.i	●	●	●	●	*SQA Records*	*11.19*	②	②	②	②
3	*Assurance is obtained that software life cycle processes comply with approved software standards.*	MB.8.1.b	MB.8.2.a MB.8.2.c MB.8.2.d MB.8.2.f MB.8.2.h MB.8.2.i	●	●	●		*SQA Records*	*11.19*	②	②	②	
4	*Assurance is obtained that transition criteria for the software life cycle processes are satisfied.*	MB.8.1.c	MB.8.2.e MB.8.2.h MB.8.2.i	●	●	●		*SQA Records*	*11.19*	②	②	②	
5	*Assurance is obtained that software conformity review is conducted.*	MB.8.1.d	MB.8.2.g MB.8.2.h *8.3*	●	●	●	●	*SQA Records*	*11.19*	②	②	②	②

Table MB.A-10 Certification Liaison Process

	Objective		Activity	Applicability by Software Level				Output		Control Category by Software Level			
	Description	Ref	Ref	A	B	C	D	Data Item	Ref	A	B	C	D
1	*Communication and understanding between the applicant and the certification authority is established.*	MB.9.0.a	MB.9.1.b MB.9.1.c	❍	❍	❍	❍	*Plan for Software Aspects of Certification*	MB.11.1	①	①	①	①
2	*The means of compliance is proposed and agreement with the Plan for Software Aspects of Certification is obtained.*	MB.9.0.b	MB.9.1.a MB.9.1.b MB.9.1.c	❍	❍	❍	❍	*Plan for Software Aspects of Certification*	MB.11.1	①	①	①	①
3	*Compliance substantiation is provided.*	MB.9.0.c	MB.9.2.a MB.9.2.b MB.9.2.c	❍	❍	❍	❍	*Software Accomplishment Summary* *Software Configuration Index*	*11.20* MB.11.16	① ①	① ①	① ①	① ①

ANNEX MB.B - ACRONYMS AND GLOSSARY OF TERMS

The acronyms and glossary definitions provided in this annex are for the terms used in this supplement. The terms defined in Annex B of DO-178C are not repeated here.

Acronym	Meaning
AL	Assurance Level
CC1	Control Category 1
CC2	Control Category 2
CNS/ATM	Communication, Navigation, Surveillance, and Air Traffic Management
COTS	Commercial Off-The-Shelf
DP	Discussion Paper
FAQ	Frequently Asked Question
HLR	High-Level Requirement
LLR	Low-Level Requirement
MB	Model-Based Development and Verification
PSAA	Plan for Software Aspects of Approval
PSAC	Plan for Software Aspects of Certification
SCM	Software Configuration Management
SDP	Software Development Plan
SQA	Software Quality Assurance
SRATS	System Requirements Allocated To Software
SVP	Software Verification Plan
SW	Software

GLOSSARY

Design Model – A model that defines any software design such as low-level requirements, software architecture, algorithms, component internal data structures, data flow and/or control flow. A model used to generate Source Code is a Design Model.

Model - An abstract representation of a given set of aspects of a system that is used for analysis, verification, simulation, code generation, or any combination thereof. A model should be unambiguous, regardless of its level of abstraction.

Note 1: If the representation is a diagram that is ambiguous in its interpretation, this is not considered to be a model.

Note 2: The "given set of aspects of a system" may contain all aspects of the system or only a subset.

Model-Based Development and Verification – A technology in which models represent software requirements and/or software design descriptions to support the software development and verification processes.

Model coverage analysis – An analysis that determines which requirements expressed by the Design Model were not exercised by verification based on the requirements from which the Design Model was developed. The purpose of this analysis is to support the detection of unintended function in the Design Model, where coverage of the requirements from which the model was developed has been achieved by the verification cases.

Model element – A unit from which a model is constructed.

Model element library – A collection of model elements used as a baseline to construct a model. A model may or may not be developed using model element libraries.

Model simulation – The activity of exercising the behavior of a model using a model simulator.

Model simulator – A device, computer program or system that enables the execution of a model to demonstrate its behavior in support of verification and/or validation.

Note: The model simulator may or may not be executing code that is representative of the target code.

Modeling technique – A modeling technique is a combination of a modeling language and a particular manner of using this modeling language. It is driven by the level of abstraction of the information to be represented by the model and the selected modeling tools.

Specification Model – A model representing high-level requirements that provides an abstract representation of functional, performance, interface, or safety characteristics of software components. A Specification Model does not define software design details such as internal data structures, internal data flow, or internal control flow.

Symbology – The graphical appearance of modeling elements. Some modeling environments allow custom symbology to be defined.

Note: This term is only used in Appendix B which is not considered "normative" but it is included here for completeness.

ANNEX MB.C – PROCESS OBJECTIVES AND OUTPUTS BY ASSURANCE LEVEL IN DO-278A

This annex provides tables that describe the objectives, related activities, and outputs given in DO-278A Annex A relevant when using model-based development and verification in the production of software for Communication, Navigation, Surveillance, and Air Traffic Management (CNS/ATM) systems. The tables include guidance for:

a. The process objectives applicable for assurance levels (AL) 1-5, as defined in DO-278A section 2.3.2.

b. The independence by assurance level of the software life cycle process activities applicable to satisfy that process's objectives.

c. The control category by assurance level for the software life cycle data produced by the software life cycle process activities (section MB.7.3).

The tables include references as appropriate to DO-278A and to this supplement. For clear identification, references to DO-278A do not include a prefix, whereas references to this supplement include "MB." as a prefix.

These tables should not be used as a checklist. In order to fully understand the guidance, the full body of DO-278A and this supplement should be considered. When reviewing this supplement for CNS/ATM systems, the following terms specific to the airborne community should be replaced as follows:

- "airborne" with "CNS/ATM system"
- "certification" with "approval"
- "airworthiness requirements" with "applicable approval requirements"
- "certification liaison process" with "approval process"
- "Plan for Software Aspects of Certification" (PSAC) with "Plan for Software Aspects of Approval" (PSAA)

This supplement does not provide guidance for the use of model-based development and verification for commercial off-the-shelf (COTS) software or adaptation parameter data (DO-278A sections 12.4 and 12.5).

The following legend applies to "Applicability by Assurance Level" and "Control Category by Assurance Level" for all tables:

LEGEND:	●	The objective should be satisfied with independence.
	❍	The objective should be satisfied.
	Blank	Satisfaction of objective is at applicant's discretion.
	①	Data satisfies the objectives of Control Category 1 (CC1).
	②	Data satisfies the objectives of Control Category 2 (CC2).

Table MB.C-1 Software Planning Process

	Objective		Activity	Applicability by Assurance Level					Output		Control Category by Assurance Level				
	Description	Ref	Ref	AL 1	AL 2	AL 3	AL 4	AL 5	Data Item	Ref	AL 1	AL 2	AL 3	AL 4	AL 5
1	*The activities of the software life cycle processes are defined.*	MB.4.1.a	MB.4.2.a MB.4.2.c MB.4.2.d MB.4.2.e MB.4.2.g MB.4.2.i MB.4.2.l MB.4.2.m MB.4.2.n MB.4.2.o MB.4.3.c	❍	❍	❍	❍	❍	*PSAA* *SDP* *SVP* *SCM Plan* *SQA Plan*	MB.11.1 MB.11.2 MB.11.3 MB.11.4 *11.5*	① ① ① ① ① ① ① ② ② ② ① ① ② ② ② ① ① ② ② ② ① ① ② ② ②				
2	*The software life cycle(s), including the inter-relationships between the processes, their sequencing, feedback mechanisms, and transition criteria, is defined.*	MB.4.1.b	MB.4.2.i MB.4.3.b	❍	❍	❍	❍		*PSAA* *SDP* *SVP* *SCM Plan* *SQA Plan*	MB.11.1 MB.11.2 MB.11.3 MB.11.4 *11.5*	① ① ① ① ① ① ② ② ① ① ② ② ① ① ② ② ① ① ② ②				
3	*Software life cycle environment is selected and defined.*	MB.4.1.c	*4.4.1* MB.4.4.2.a MB.4.4.2.b MB.4.4.2.c MB.4.4.3 MB.4.4.4	❍	❍	❍	❍		*PSAA* *SDP* *SVP* *SCM Plan* *SQA Plan*	MB.11.1 MB.11.2 MB.11.3 MB.11.4 *11.5*	① ① ① ① ① ① ② ② ① ① ② ② ① ① ② ② ① ① ② ②				
4	*Additional considerations are addressed.*	MB.4.1.d	MB.4.2.f MB.4.2.h MB.4.2.i MB.4.2.j MB.4.2.	❍	❍	❍	❍	❍	*PSAA* *SDP* *SVP* *SCM Plan* *SQA Plan*	MB.11.1 MB.11.2 MB.11.3 MB.11.4 *11.5*	① ① ① ① ① ① ① ② ② ② ① ① ② ② ② ① ① ② ② ② ① ① ② ② ②				
5	*Software development standards are defined.*	MB.4.1.e	MB.4.2.b MB.4.2.g MB.4.5	❍	❍	❍	❍		*SW Requirements Standards* *SW Design Standards* *SW Code Standards* SW Model Standards	*11.6* *11.7* *11.8* MB.11.23	① ① ② ② ① ① ② ② ① ① ② ② ① ① ② ②				
6	*Software plans comply with this document.*	MB.4.1.f	MB.4.3.a MB.4.6	❍	❍	❍	❍		*Software Verification Results*	MB.11.14	② ② ② ②				
7	*Development and revision of software plans are coordinated.*	MB.4.1.g	MB.4.2.g MB.4.6	❍	❍	❍	❍		*Software Verification Results*	MB.11.14	② ② ② ②				

Table Mb.C-2 Software Development Processes

	Objective		Activity	Applicability by Assurance Level					Output		Control Category by Assurance Level				
	Description	Ref	Ref	AL 1	AL 2	AL 3	AL 4	AL 5	Data Item	Ref	AL 1	AL 2	AL 3	AL 4	AL 5
1	*High-level requirements are developed.*	MB.5.1.1.a	MB.5.1.2.a MB.5.1.2.b MB.5.1.2.c MB.5.1.2.d MB.5.1.2.e MB.5.1.2.f MB.5.1.2.g MB.5.1.2.j MB.5.1.2.k MB.5.2.1.l MB.5.5 *5.5.a*	❍	❍	❍	❍	❍	*Software Requirements Data* *Trace Data*	MB.11.9 MB.11.21	① ①	① ①	① ①	① ①	① ①
2	*Derived high-level requirements are defined and provided to the system processes, including the system safety assessment process.*	MB.5.1.1.b	MB.5.1.2.h MB.5.1.2.i MB.5.1.2.k	❍	❍	❍	❍	❍	*Software Requirements Data*	MB.11.9	①	①	①	①	①
3	*Software architecture is developed.*	MB.5.2.1.a	MB.5.2.2.a MB.5.2.2.d MB.5.2.2.h	❍	❍	❍	❍	❍	*Design Description*	MB.11.10	①	①	①	①	②
4	*Low-level requirements are developed.*	MB.5.2.1.a	MB.5.2.2.a MB.5.2.2.e MB.5.2.2.f MB.5.2.2.g MB.5.2.2.h MB.5.2.3.a MB.5.2.3.b *5.2.4.a* *5.2.4.b* *5.2.4.c* MB.5.5 *5.5.b*	❍	❍	❍			*Design Description* *Trace Data*	MB.11.10 MB.11.21	① ①	① ①	① ①		
5	*Derived low-level requirements are defined and provided to the system processes, including the system safety assessment process.*	MB.5.2.1.b	MB.5.2.2.b MB.5.2.2.c MB.5.2.2.h	❍	❍	❍			*Design Description*	MB.11.10	①	①	①		
6	*Source Code is developed.*	*5.3.1.a*	*5.3.2.a* *5.3.2.b* *5.3.2.c* *5.3.2.d* MB.5.5 *5.5.c*	❍	❍	❍			*Source Code* *Trace Data*	*11.11* MB.11.21	① ①	① ①	① ①		

Table MB.C-2 (Continued) Software Development Processes

	Objective		Activity	Applicability by Assurance Level					Output		Control Category by Assurance Level				
	Description	Ref	Ref	AL 1	AL 2	AL 3	AL 4	AL 5	Data Item	Ref	AL 1	AL 2	AL 3	AL 4	AL 5
7	*Executable Object Code and Adaptation Data Item Files, if any, are produced and loaded in the target computer.*	*5.4.1.a*	5.4.2.a 5.4.2.b 5.4.2.c 5.4.2.d 5.4.2.e 5.4.2.f	❍	❍	❍	❍	❍	*Executable Object Code* *Adaptation Data Item File*	*11.12* *11.22*	① ①	① ①	① ①	① ①	① ①
MB 8	Specification Model elements that do not contribute to implementation or realization of any high-level requirement are identified.	MB.5.1.1.c	MB.5.1.2.k	❍	❍	❍	❍	❍	Software Requirements Data	MB.11.9	①	①	①	①	①
MB 9	Design Model elements that do not contribute to implementation or realization of any software architecture are identified.	MB.5.2.1.c	MB.5.2.2.h	❍	❍	❍	❍	❍	Design Description	MB.11.10	①	①	①	①	②
MB 10	Design Model elements that do not contribute to implementation or realization of any low-level requirement are identified.	MB.5.2.1.c	MB.5.2.2.h	❍	❍	❍			Design Description	MB.11.10	①	①	①		

Table Mb.C-3 Verification of Outputs of Software Requirements Process

	Objective		Activity	Applicability by Assurance Level					Output		Control Category by Assurance Level				
	Description	Ref	Ref	AL 1	AL 2	AL 3	AL 4	AL 5	Data Item	Ref	AL 1	AL 2	AL 3	AL 4	AL 5
1	*High-level requirements comply with system requirements.* (See Item 1)	MB.6.3.1.a	MB.6.3.1 MB.6.8.1 (See Item 2)	●	●	❍	❍	❍	*Software Verification Results*	MB.11.14	②	②	②	②	②
2	*High-level requirements are accurate and consistent.*	MB.6.3.1.b	MB.6.3.1 MB.6.8.1 (See Item 2)	●	●	❍	❍	❍	*Software Verification Results*	MB.11.14	②	②	②	②	②
3	*High-level requirements are compatible with target computer.*	MB.6.3.1.c	MB.6.3.1	❍	❍				*Software Verification Results*	MB.11.14	②	②			
4	*High-level requirements are verifiable.*	MB.6.3.1.d	MB.6.3.1 MB.6.8.1 (See Item 2)	❍	❍	❍	❍		*Software Verification Results*	MB.11.14	②	②	②	②	
5	*High-level requirements conform to standards.*	MB.6.3.1.e	MB.6.3.1	❍	❍	❍	❍		*Software Verification Results*	MB.11.14	②	②	②	②	
6	*High-level requirements are traceable to system requirements.* (See Item 1)	MB.6.3.1.f	MB.6.3.1	❍	❍	❍	❍	❍	*Software Verification Results*	MB.11.14	②	②	②	②	②
7	*Algorithms are accurate.*	MB.6.3.1.g	MB.6.3.1 MB.6.8.1 (See Item 2)	●	●	❍	❍		*Software Verification Results*	MB.11.14	②	②	②	②	
MB 8	*Simulation cases are correct.* (See Item 2)	MB.6.8.3.2.a	MB.6.8.1 MB.6.8.3.2	●	❍	❍	❍	❍	*Software Verification Results*	MB.11.14	②	②	②	②	②
MB 9	*Simulation procedures are correct.* (See Item 2)	MB.6.8.3.2.b	MB.6.8.1 MB.6.8.3.2	●	❍	❍	❍	❍	*Software Verification Results*	MB.11.14	②	②	②	②	②
MB 10	*Simulation results are correct and discrepancies explained.* (See Item 2)	MB.6.8.3.2.c	MB.6.8.1 MB.6.8.3.2	●	❍	❍	❍	❍	*Software Verification Results*	MB.11.14	②	②	②	②	②

Item 1: In case the requirements from which a Design Model is developed are output of the system process, the objectives 1 and 6 are implicitly satisfied for these requirements.

Item 2: As described in section MB.6.8.1 of the supplement, simulation may be used as a means of compliance for objectives 1, 2, 4 or 7 of this table. If simulation is used as this means, objectives MB.8, MB.9, and MB.10 are required.

Table MB.C-4 Verification of Outputs of Software Design Process

	Objective		Activity	Applicability by Assurance Level					Output		Control Category by Assurance Level				
	Description	Ref	Ref	AL 1	AL 2	AL 3	AL 4	AL 5	Data Item	Ref	AL 1	AL 2	AL 3	AL 4	AL 5
1	*Low-level requirements comply with high-level requirements.*	MB.6.3.2.a	MB.6.3.2 MB.6.7 MB.6.8.1 (See Item 1)	●	●	❍			*Software Verification Results*	MB.11.14	②	②	②		
2	*Low-level requirements are accurate and consistent.*	MB.6.3.2.b	MB.6.3.2 MB.6.8.1 (See Item 1)	●	●	❍			*Software Verification Results*	MB.11.14	②	②	②		
3	*Low-level requirements are compatible with target computer.*	MB.6.3.2.c	MB.6.3.2	❍	❍				*Software Verification Results*	MB.11.14	②	②			
4	*Low-level requirements are verifiable.*	MB.6.3.2.d	MB.6.3.2 MB.6.8.1 (See Item 1)	❍	❍				*Software Verification Results*	MB.11.14	②	②			
5	*Low-level requirements conform to standards.*	MB.6.3.2.e	MB.6.3.2	❍	❍	❍			*Software Verification Results*	MB.11.14	②	②	②		
6	*Low-level requirements are traceable to high-level requirements.*	MB.6.3.2.f	MB.6.3.2	❍	❍	❍			*Software Verification Results*	MB.11.14	②	②	②		
7	*Algorithms are accurate.*	MB.6.3.2.g	MB.6.3.2 MB.6.8.1 (See Item 1)	●	●	❍	❍		*Software Verification Results*	MB.11.14	②	②	②	②	
8	*Software architecture is compatible with high-level requirements.*	MB.6.3.3.a	MB.6.3.3 MB.6.8.1 (See Item 1)	●	❍	❍	❍		*Software Verification Results*	MB.11.14	②	②	②	②	
9	*Software architecture is consistent.*	MB.6.3.3.b	MB.6.3.3 MB.6.8.1 (See Item 1)	●	❍	❍	❍		*Software Verification Results*	MB.11.14	②	②	②	②	
10	*Software architecture is compatible with target computer.*	MB.6.3.3.c	MB.6.3.3	❍	❍				*Software Verification Results*	MB.11.14	②	②			
11	*Software architecture is verifiable.*	MB.6.3.3.d	MB.6.3.3 MB.6.8.1 (See Item 1)	❍	❍				*Software Verification Results*	MB.11.14	②	②			
12	*Software architecture conforms to standards.*	MB.6.3.3.e	MB.6.3.3	❍	❍	❍	❍		*Software Verification Results*	MB.11.14	②	②	②	②	
13	*Software partitioning integrity is confirmed.*	MB.6.3.3.f	MB.6.3.3	●	❍	❍	❍	❍	*Software Verification Results*	MB.11.14	②	②	②	②	②
MB 14	*Simulation cases are correct.* (See Item 1)	MB.6.8.3.2.a	MB.6.8.1 MB.6.8.3.2	●	❍	❍	❍		*Software Verification Results*	MB.11.14	②	②	②	②	

Table MB.C-4 (Continued) Verification of Outputs of Software Design Process

	Objective		Activity	Applicability by Assurance Level					Output		Control Category by Assurance Level				
	Description	Ref	Ref	AL 1	AL 2	AL 3	AL 4	AL 5	Data Item	Ref	AL 1	AL 2	AL 3	AL 4	AL 5
MB 15	*Simulation procedures are correct.* (See Item 1)	MB.6.8.3.2.b	MB.6.8.1 MB.6.8.3.2	●	❍	❍	❍		*Software Verification Results*	MB.11.14	②	②	②	②	
MB 16	*Simulation results are correct and discrepancies explained.* (See Item 1)	MB.6.8.3.2.c	MB.6.8.1 MB.6.8.3.2	●	❍	❍	❍		*Software Verification Results*	MB.11.14	②	②	②	②	

Item 1: As described in section MB.6.8.1 of this supplement, simulation may be used as a means of compliance for objectives 1, 2, 4, 7, 8, 9, or 11 of this table. If simulation is used as this means, objectives MB.14, MB.15, and MB.16 are required.

Table MB.C-5 Verification of Outputs of Software Coding & Integration Processes

	Objective		Activity	Applicability by Assurance Level					Output		Control Category by Assurance Level				
	Description	Ref	Ref	AL 1	AL 2	AL 3	AL 4	AL 5	Data Item	Ref	AL 1	AL 2	AL 3	AL 4	AL 5
1	*Source Code complies with low-level requirements.*	MB.6.3.4.a	MB.6.3.4	●	●	❍			*Software Verification Results*	MB.11.14	②	②	②		
2	*Source Code complies with software architecture.*	MB.6.3.4.b	MB.6.3.4	●	❍	❍			*Software Verification Results*	MB.11.14	②	②	②		
3	*Source Code is verifiable.*	MB.6.3.4.c	MB.6.3.4	❍	❍				*Software Verification Results*	MB.11.14	②	②			
4	*Source Code conforms to standards.*	MB.6.3.4.d	MB.6.3.4	❍	❍	❍			*Software Verification Results*	MB.11.14	②	②	②		
5	*Source Code is traceable to low-level requirements.*	MB.6.3.4.e	MB.6.3.4	❍	❍	❍			*Software Verification Results*	MB.11.14	②	②	②		
6	*Source Code is accurate and consistent.*	MB.6.3.4.f	MB.6.3.4	●	❍	❍			*Software Verification Results*	MB.11.14	②	②	②		
7	*Output of software integration process is complete and correct.*	*6.3.5.a*	*6.3.5*	❍	❍	❍	❍		*Software Verification Results*	MB.11.14	②	②	②	②	
8	*Adaptation Data Item File is correct and complete.*	*6.6.a*	*6.6*	●	●	❍	❍	❍	*Software Verification Cases and Procedures* *Software Verification Results*	MB.11.13 MB.11.14	① ②	① ②	② ②	② ②	② ②
9	*Verification of Adaptation Data Item File is achieved*	*6.6.b*	*6.6*	●	●	❍	❍		*Software Verification Results*	MB.11.14	②	②	②	②	

Table Mb.C-6 Testing of Outputs of Integration Process

	Objective		Activity	Applicability by Assurance Level					Output		Control Category by Assurance Level				
	Description	Ref	Ref	AL 1	AL 2	AL 3	AL 4	AL 5	Data Item	Ref	AL 1	AL 2	AL 3	AL 4	AL 5
1	*Executable Object Code complies with high-level requirements.*	*6.4.a*	*6.4.2* *6.4.2.1* *6.4.3* *6.5* MB.6.8.2.a (See Item 1)	❍	❍	❍	❍	❍	*Software Verification Cases and Procedures* *Software Verification Results* *Trace Data*	MB.11.13 MB.11.14 MB.11.21	① ② ①	① ② ①	② ② ②	② ② ②	② ② ②
2	*Executable Object Code is robust with high-level requirements.*	*6.4.b*	*6.4.2* *6.4.2.2* *6.4.3* *6.5* MB.6.8.2.a (See Item 1)	❍	❍	❍	❍	❍	*Software Verification Cases and Procedures* *Software Verification Results* *Trace Data*	MB.11.13 MB.11.14 MB.11.21	① ② ①	① ② ①	② ② ②	② ② ②	② ② ②
3	*Executable Object Code complies with low-level requirements.*	*6.4.c*	*6.4.2* *6.4.2.1* *6.4.3* *6.5*	●	●	❍			*Software Verification Cases and Procedures* *Software Verification Results* *Trace Data*	MB.11.13 MB.11.14 MB.11.21	① ② ①	① ② ①	② ② ②		
4	*Executable Object Code is robust with low-level requirements.*	*6.4.d*	*6.4.2* *6.4.2.2* *6.4.3* *6.5*	●	❍	❍			*Software Verification Cases and Procedures* *Software Verification Results* *Trace Data*	MB.11.13 MB.11.14 MB.11.21	① ② ①	① ② ①	② ② ②		
5	*Executable Object Code is compatible with target computer.*	*6.4.e*	*6.4.1.a* *6.4.3.a*	❍	❍	❍	❍	❍	*Software Verification Cases and Procedures* *Software Verification Results*	MB.11.13 MB.11.14	① ②	① ②	② ②	② ②	② ②

Item 1: As described in section MB.6.8.2.a of the supplement, the MB.6.8.2.a activity is only required when simulation is used as a means of compliance for objectives 1 or 2 of this table.

Table MB.C-7 Verification of Verification Process Results

	Objective		Activity	Applicability by Assurance Level					Output		Control Category by Assurance Level				
	Description	Ref	Ref	AL 1	AL 2	AL 3	AL 4	AL 5	Data Item	Ref	AL 1	AL 2	AL 3	AL 4	AL 5
1	*Test procedures are correct.*	*6.4.5.b*	*6.4.5*	●	❍	❍	❍		*Software Verification Results*	MB.11.14	②	②	②	②	
2	*Test results are correct and discrepancies explained.*	*6.4.5.c*	*6.4.5*	●	❍	❍	❍		*Software Verification Results*	MB.11.14	②	②	②	②	
3	*Test coverage of high-level requirements is achieved.*	*6.4.4.a*	*6.4.4.1* MB.6.8.2.a	●	❍	❍	❍	❍	*Software Verification Results*	MB.11.14	②	②	②	②	②
4	*Test coverage of low-level requirements is achieved.*	*6.4.4.b*	*6.4.4.1* MB.6.7	●	❍	❍			*Software Verification Results*	MB.11.14	②	②	②		
5	*Test coverage of software structure (modified condition/decision) is achieved.*	*6.4.4.c*	*6.4.4.2.a* *6.4.4.2.b* *6.4.4.2.d* *6.4.4.3* MB.6.8.3.2.b (See Item 1)	●					*Software Verification Results*	MB.11.14	②				
6	*Test coverage of software structure (decision coverage) is achieved.*	*6.4.4.c*	*6.4.4.2.a* *6.4.4.2.b* *6.4.4.2.d* *6.4.4.3* MB.6.8.3.2.b (See Item 1)	●	●				*Software Verification Results*	MB.11.14	②	②			
7	*Test coverage of software structure (statement coverage) is achieved.*	*6.4.4.c*	*6.4.4.2.a* *6.4.4.2.b* *6.4.4.2.d* *6.4.4.3* MB.6.8.3.2.b (See Item1)	●	●	❍			*Software Verification Results*	MB.11.14	②	②	②		
8	*Test coverage of software structure (data coupling and control coupling) is achieved.*	*6.4.4.d*	*6.4.4.2.c* *6.4.4.2.d* *6.4.4.3* MB.6.8.3.2.b (See Item 1)	●	●	❍	❍		*Software Verification Results*	MB.11.14	②	②	②	②	
9	*Verification of additional code, that cannot be traced to Source Code, is achieved.*	*6.4.4.c*	*6.4.4.2.b*	●					*Software Verification Results*	MB.11.14	②				
MB 10	Simulation cases are correct. (See Item 2)	MB.6.8.3.2.a	MB.6.8.3.2	●	❍	❍	❍		*Software Verification Results*	MB.11.14	②	②	②	②	

Table MB.C-7 (Continued) Verification of Verification Process Results

	Objective		Activity	Applicability by Assurance Level					Output		Control Category by Assurance Level				
	Description	Ref	Ref	AL 1	AL 2	AL 3	AL 4	AL 5	Data Item	Ref	AL 1	AL 2	AL 3	AL 4	AL 5
MB 11	Simulation procedures are correct. (See Item 2)	MB.6.8.3.2.b	MB.6.8.3.2	●	❍	❍	❍		*Software Verification Results*	MB.11.14	②	②	②	②	
MB 12	Simulation results are correct and discrepancies are explained. (See Item 2)	MB.6.8.3.2.c	MB.6.8.3.2	●	❍	❍	❍		*Software Verification Results*	MB.11.14	②	②	②	②	

Item 1: As described in section MB.6.8.2 of the supplement, the MB.6.8.2.b activity is only required when simulation is used as a means of compliance for any objectives 5, 6, 7, or 8 of this table.

Item 2: As described in section MB.6.8.2 of this supplement, these three objectives are only required when simulation is used as a means of compliance for objectives 1 or 2 of Annex Table MB.A-6.

Table Mb.C-8 Software Configuration Management Process

	Objective		Activity	Applicability by Assurance Level					Output		Control Category by Assurance Level				
	Description	Ref	Ref	AL 1	AL 2	AL 3	AL 4	AL 5	Data Item	Ref	AL 1	AL 2	AL 3	AL 4	AL 5
1	*Configuration items are identified.*	MB.7.1.a	*7.2.1*	❍	❍	❍	❍	❍	*SCM Records*	MB.11.18	②	②	②	②	②
2	*Baselines and traceability are established.*	MB.7.1.b	MB.7.2.2	❍	❍	❍	❍	❍	*Software Configuration Index* *SCM Records*	MB.11.16 MB.11.18	② ②	② ②	② ②	② ②	② ②
3	*Problem reporting, change control, change review, and configuration status accounting are established.*	MB.7.1.c MB.7.1.d MB.7.1.e MB.7.1.f	*7.2.3* *7.2.4* *7.2.5* *7.2.6*	❍	❍	❍	❍	❍	*Problem Reports* *SCM Records*	*11.17* MB.11.18	② ②	② ②	② ②	② ②	② ②
4	*Archive, retrieval, and release are established.*	MB.7.1.g	*7.2.7*	❍	❍	❍	❍	❍	*SCM Records*	MB.11.18	②	②	②	②	②
5	*Software load control is established.*	MB.7.1.h	*7.4*	❍	❍	❍	❍	❍	*SCM Records*	MB.11.18	②	②	②	②	②
6	*Software life cycle environment control is established.*	MB.7.1.i	MB.7.5	❍	❍	❍	❍	❍	*Software Life Cycle Environment* *SCM Records*	MB.11.15 MB.11.18	② ②	② ②	② ②	② ②	② ②

Table MB.C-9 Software Quality Assurance Process

	Objective		Activity	Applicability by Assurance Level					Output		Control Category by Assurance Level				
	Description	Ref	Ref	AL 1	AL 2	AL 3	AL 4	AL 5	Data Item	Ref	AL 1	AL 2	AL 3	AL 4	AL 5
1	*Assurance is obtained that software plans and standards are developed and reviewed for compliance to* DO-278A and this Supplement *and for consistency.*	MB.8.1.a	MB.8.2.b MB.8.2.h MB.8.2.i	●	●	●	●		*SQA Records*	*11.19*	②	②	②	②	
2	*Assurance is obtained that software life cycle processes comply with approved software plans.*	MB.8.1.b	MB.8.2.a MB.8.2.c MB.8.2.d MB.8.2.f MB.8.2.h MB.8.2.i	●	●	●	●	●	*SQA Records*	*11.19*	②	②	②	②	②
3	*Assurance is obtained that software life cycle processes comply with approved software standards.*	MB.8.1.b	MB.8.2.a MB.8.2.c MB.8.2.d MB.8.2.f MB.8.2.h MB.8.2.i	●	●	●	●		*SQA Records*	*11.19*	②	②	②	②	
4	*Assurance is obtained that transition criteria for the software life cycle processes are satisfied.*	MB.8.1.c	MB.8.2.e MB.8.2.h MB.8.2.i	●	●	●	●		*SQA Records*	*11.19*	②	②	②	②	
5	*Assurance is obtained that software conformity review is conducted.*	MB.8.1.d	MB.8.2.g MB.8.2.h *8.3*	●	●	●	●	●	*SQA Records*	*11.19*	②	②	②	②	②

Table MB.C-10 Software Approval Process

	Objective		Activity	Applicability by Assurance Level					Output		Control Category by Assurance Level				
	Description	Ref	Ref	AL 1	AL 2	AL 3	AL 4	AL 5	Data Item	Ref	AL 1	AL 2	AL 3	AL 4	AL 5
1	*Communication and understanding between the applicant and the approval authority is established.*	MB.9.0.a	MB.9.1.b MB.9.1.c	❍	❍	❍	❍	❍	*Plan for Software Aspects of Approval*	MB.11.1	①	①	①	①	①
2	*The means of compliance is proposed and agreement with the Plan for Software Aspects of Approval is obtained.*	MB.9.0.b	MB.9.1.a MB.9.1.b MB.9.1.c	❍	❍	❍	❍	❍	*Plan for Software Aspects of Approval*	MB.11.1	①	①	①	①	①
3	*Compliance substantiation is provided.*	MB.9.0.c	MB.9.2.a MB.9.2.b MB.9.2.c	❍	❍	❍	❍	❍	*Software Accomplishment Summary* *Software Configuration Index*	*11.20* MB.11.16	① ①	① ①	① ①	① ①	① ①

APPENDIX MB.A – COMMITTEE MEMBERSHIP

Executive Committee Members

Jim Krodel, Pratt & Whitney	SC-205 Chair
Gérard Ladier, Airbus/Aerospace Valley	WG-71 Chair
Mike DeWalt, Certification Services, Inc./FAA	SC-205 Secretary (until March 2008)
Leslie Alford, Boeing Company	SC-205 Secretary (from March 2008)
Ross Hannan, Sigma Associates (Aerospace)	WG-71 Secretary
Barbara Lingberg, FAA	FAA Representative/CAST Chair
Jean-Luc Delamaide, EASA	EASA Representative
John Coleman, Dawson Consulting	Sub-group Liaison
Matt Jaffe, Embry-Riddle Aeronautical University	Web Site Liaison
Todd R. White, L-3 Communications/Qualtech	Collaborative Technology Software Liaison

Sub-Group Leadership

SG-1 – Document Integration

Ron Ashpole, SILVER ATENA	SG-1 Co-chair
Tom Ferrell, Ferrell and Associates Consulting	SG-1 Co-chair (until March 2008)
Marty Gasiorowski, Worldwide Certification Services	SG-1 Co-chair (from March 2008)
Tom Roth, Airborne Software Certification Consulting	SG-1 Secretary

SG-2 – Issues and Rationale

Ross Hannan, Sigma Associates (Aerospace)	SG-2 Co-chair
Mike DeWalt, Certification Services, Inc./FAA	SG-2 Co-chair (until March 2008)
Will Struck, FAA	SG-2 Co-chair (until March 2009)
Fred Moyer, Rockwell Collins	SG-2 Co-chair (from April 2009)
John Angermayer, Mitre	SG-2 Secretary

SG-3 – Tool Qualification

Frédéric Pothon, ACG Solutions	SG-3 Co-chair
Leanna Rierson, Digital Safety Consulting	SG-3 Co-chair
Bernard Dion, Esterel Technologies	SG-3 Co-secretary
Gene Kelly, CertTech	SG-3 Co-secretary (until May 2009)
Mo Piper, Boeing Company	SG-3 Co-secretary (from May 2009)

SG-4 – Model-Based Development and Verification

Pierre Lionne, EADS APSYS	SG-4 Co-chair
Mark Lillis, Goodrich GPECS	SG-4 Co-chair
Hervé Delseny, Airbus	SG-4 Co-chair
Martha Blankenberger, Rolls-Royce	SG-4 Secretary

SG-5 – Object-Oriented Technology

Peter Heller, Airbus Operations GmbH	SG-5 Co-chair (until February 2009)
Jan-Hendrik Boelens, Eurocopter	SG-5 Co-chair (Feb 2009 to August 2010)
James Hunt, aicas	SG-5 Co-chair (from August 2010)
Jim Chelini, Verocel	SG-5 Co-chair (until Oct 2009)
Greg Millican, Honeywell	SG-5 Co-chair (Oct 2009 to August 2011)
Jim Chelini, Verocel	SG-5 Co-chair (from August 2011)

SG-6 – Formal Methods

Duncan Brown, Aero Engine Controls (Rolls-Royce)	SG-6 Co-chair
Kelly Hayhurst, NASA	SG-6 Co-chair

SG-7 – Special Considerations and CNS/ATM

David Hawken, NATS	SG-7 Co-chair (until June 2010)
Jim Stewart, NATS	SG-7 Co-chair (from June 2010)
Don Heck, Boeing Company	SG-7 Co-chair
Leslie Alford, Boeing Company	SG-7 Secretary (until March 2008)
Marguerite Baier, Honeywell	SG-7 Secretary (from March 2008)

RTCA Representative:

Rudy Ruana	RTCA Inc. (until September 2009)
Ray Glennon	RTCA Inc. (until March 2010)
Hal Moses	RTCA Inc. (until August 2010)
Cyndy Brown	RTCA Inc. (until August 2011)
Hal Moses	RTCA Inc. (from August 2011)

EUROCAE Representative:

Gilbert Amato	EUROCAE (until September 2009)
Roland Mallwitz	EUROCAE (from October 2009)

EDITORIAL COMMITTEE

Leanna Rierson, Digital Safety Consulting	Editorial Committee Chair
Ron Ashpole, SILVER ATENA	Editorial Committee
Alex Ayzenberg, Boeing Company	Editorial Committee
Patty (Bartels) Bath, Esterline AVISTA	Editorial Committee
Dewi Daniels, Verocel	Editorial Committee
Hervé Delseny, Airbus	Editorial Committee
Andrew Elliott, Design Assurance	Editorial Committee
Kelly Hayhurst, NASA	Editorial Committee
Barbara Lingberg, FAA	Editorial Committee
Steven C. Martz, Garmin	Editorial Committee
Steve Morton, TBV Associates	Editorial Committee
Marge Sonnek, Honeywell	Editorial Committee

Committee Membership

Name	Organization
Kyle Achenbach	Rolls-Royce
Dana E. Adkins	Kidde Aerospace
Leslie Alford	Boeing Company
Carlo Amalfitano	Certon Software, Inc
Gilbert Amato	EUROCAE
Peter Amey	Praxis High Integrity Systems
Allan Gilmour Anderson	Embraer
Håkan Anderwall	Saab AB
Joseph Angelo	NovAtel Inc, Canada
John Charles Angermayer	Mitre Corp
Robert Annis	GE Aviation
Ron Ashpole	SILVER ATENA
Alex Ayzenberg	Boeing Company
Marguerite Baier	Honeywell
Fred Barber	Avidyne
Clay Barber	Garmin International
Gerald F. Barofsky	L-3 Communications
Patty (Bartels) Bath	Esterline AVISTA
Brigitte Bauer	Thales
Phillipe Baufreton	SAGEM DS Safran Group
Connie Beane	ENEA Embedded Technology Inc
Bernard Beaudouin	EADS APSYS
Germain Beaulieu	Independent Consultant
Martin Beeby	Seaweed Systems
Scott Beecher	Pratt & Whitney
Haik Biglari	Fairchild Controls
Peter Billing	Aviya Technologies Inc
Denise Black	Embedded Plus Engineering
Brad Blackhurst	Independent Consultant
Craig Bladow	Woodward
Martha Blankenberger	Rolls-Royce
Holger Blasum	SYSGO
Thomas Bleichner	Rohde & Schwarz
Don Bockenfeld	CMC Electronics
Jan-Hendrik Boelens	Eurocopter
Eric Bonnafous	CommunicationSys
Jean-Christophe Bonnet	CEAT
Hugues Bonnin	Cap Gemini
Matteo Bordin	AdaCore
Feliks Bortkiewicz	Boeing
Julien Bourdeau	DND (Canada)
Paul Bousquet	Volpe National Transportation Systems Center
David Bowen	EUROCAE
Elizabeth Brandli	FAA
Andrew Bridge	EASA
Paul Brook	Thales
Daryl Brooke	Universal Avionics Systems Corporation
Duncan Brown	Aero Engine Controls (Rolls-Royce)
Thomas Buchberger	Siemens AG
Brett Burgeles	Consultant

Name	Organization
Bernard Buscail	Airbus
Bob Busser	Systems and Software Consortium
Christopher Caines	QinetiQ
Cristiano Campos Almeida De Freitas	Embraer
Jean-Louis Camus	Esterel Technologies
Richard Canis	EASA
Yann Carlier	DGAC
Luc Casagrande	EADS Apsys
Mark Chapman	Hamilton Sundstrand
Scott Chapman	FAA
Jim Chelini	Verocel
Daniel Chevallier	Thales
John Chilenski	Boeing Company
Subbiah Chockalingam	HCL Technologies
Chris Clark	Sysgo
Darren Cofer	Rockwell Collins
Keith Coffman	Goodrich
John Coleman	Dawson Consulting
Cyrille Comar	AdaCore
Ray Conrad	Lockheed Martin
Mirko Conrad	The MathWorks, Inc.
Nathalie Corbovianu	DGAC
Ana Costanti	Embraer
Dewi Daniels	Verocel
Eric Danielson	Rockwell Collins
Henri De La Vallée Poussin	SABCA
Michael Deitz	Gentex Corporation
Jean-Luc Delamaide	EASA
Hervé Delseny	Airbus
Patrick Desbiens	Transport Canada
Mike DeWalt	Certification Services, Inc./FAA
Mansur Dewshi	Ultra Electronics Controls
Bernard Dion	Esterel Technologies
Antonio Jose Vitorio Domiciano	Embraer
Kurt Doppelbauer	TTTech
Cheryl Dorsey	Digital Flight
Rick Dorsey	Digital Flight
John Doughty	Garmin International
Vincent Dovydaitis III	Foliage Software Systems, Inc.
Georges Duchein	DGA
Branimir Dulic	Transport Canada
Gilles Dulon	SAGEM DS Safran Group
Paul Dunn	Northrop Grumman Corporation
Andrew Eaton	UK CAA
Brian Eckmann	Universal Avionics Systems Corporation
Vladimir Eliseev	Sukhoi Civil Aircraft Company (SCAC)
Andrew Elliott	Design Assurance
Mike Elliott	Boeing Company
Joao Esteves	Critical Software
Rowland Evans	Pratt & Whitney Canada
Louis Fabre	Eurocopter

Name	Organization
Martin Fassl	Siemens AG
Michael Fee	Aero Engine Controls (Rolls-Royce)
Tom Ferrell	Ferrell and Associates Consulting
Uma Ferrell	Ferrell and Associates Consulting
Lou Fisk	GE Aviation
Ade Fountain	Penny and Giles
Claude Fournier	Liebherr
Pierre Francine	Thales
Timothy Frey	Honeywell
Stephen J. Fridrick	GE Aviation
Leonard Fulcher	TTTech
Randall Fulton	Seaweed Systems
Francoise Gachet	Dassault-Aviation
Victor Galushkin	GosNIIAS
Marty Gasiorowski	Worldwide Certification Services
Stephanie Gaudan	Thales
Jean-Louis Gebel	Airbus
Dries Geldof	BARCO
Dimitri Gianesini	Airbus
Jim Gibbons	Boeing Company
Dara Gibson	FAA
Greg Gicca	AdaCore
Steven Gitelis	Lumina Engineering
Ian Glazebrook	WS Atkins
Santiago Golmayo	GMV SA
Ben Gorry	British Aerospace Systems
Florian Gouleau	DGA Techniques Aéronautiques
Olivier Graff	Intertechnique - Zodiac
Russell DeLoy Graham	Garmin International
Robert Green	BAE Systems
Mark Grindle	Systems Enginuity
Peter Grossinger	Pilatus Aircraft
Mark Gulick	Solers, Inc.
Pierre Guyot	Dassault Aviation
Ibrahim Habli	University of York
Ross Hannan	Sigma Associates (Aerospace) Limited
Christopher H. Hansen	Rockwell Collins
Wue Hao Wen	Civil Aviation Administration of China (CAAC)
Keith Harrison	HVR Consulting Services Ltd
Bjorn Hasselqvist	Saab AB
Kevin Hathaway	Aero Engine Controls (Goodrich)
David Hawken	NATS
Kelly Hayhurst	NASA
Peter Heath	Securaplane Technologies
Myron Hecht	Aerospace Corporation
Don Heck	Boeing Company
Peter Heller	Airbus Operations GmbH
Barry Hendrix	Lockheed Martin
Michael Hennell	LDRA
Michael Herring	Rockwell Collins
Ruth Hirt	FAA

Name	Organization
Kent Hollinger	Mitre Corp
C. Michael Holloway	NASA
Ian Hopkins	Aero Engine Controls (Rolls-Royce)
Gary Horan	FAA
Chris Hote	PolySpace Inc.
Susan Houston	FAA
James Hummell	Embedded Plus
Dr. James J. Hunt	aicas
Rebecca L. Hunt	Boeing Company
Stuart Hutchesson	Aero Engine Controls (Rolls-Royce)
Rex Hyde	Moog Inc. Aircraft Group
Mario Iacobelli	Mannarino Systems
Melissa Isaacs	FAA
Vladimir Istomin	Sukhoi Civil Aircraft Company (SCAC)
Stephen A. Jacklin	NASA
Matt Jaffe	Embry-Riddle Aeronautical University
Marek Jaglarz	Pilatus Aircraft
Myles Jalalian	FAA
Merlin James	Garmin International
Tomas Jansson	Saab AB
Eric Jenn	Thales
Lars Johannknecht	EADS
Rikard Johansson	Saab AB
John Jorgensen	Universal Avionics Systems
Jeffrey Joyce	Critical Systems Labs
Chris Karis	Ensco
Gene Kelly	CertTech
Anne-Cécile Kerbrat	Aeroconseil
Randy Key	FAA
Charles W. Kilgore II	FAA
Wayne King	Honeywell
Daniel Kinney	Boeing Company
Judith Klein	Lockheed Martin
Joachim Klichert	Diehl Avionik Systeme
Jeff Knickerbocker	Sunrise Certification & Consulting, Inc.
John Knight	University of Virginia
Rainer Kollner	Verocel
Andrew Kornecki	Embry-Riddle Aeronautical University
Igor Koverninskiy	Gos NIIAS
Jim Krodel	Pratt & Whitney
Paramesh Kunda	Pratt & Whitney Canada
Sylvie Lacabanne	AIRBUS
Gérard Ladier	Airbus/Aerospace Valley
Ron Lambalot	Boeing Company
Boris Langer	Diehl Aerospace
Susanne Lanzerstorfer	APAC GesmbH
Gilles Laplane	SAGEM DS Safran Group
Jeanne Larsen	Hamilton Sundstrand
Emmanuel Ledinot	Dassault Aviation
Stephane Leriche	Thales
Hong Leung	Bell Helicopter Textron

Name	Organization
John Lewis	FAA
John Li	Thales
Mark Lillis	Goodrich GPECS
Barbara Lingberg	FAA
Pierre Lionne	EADS APSYS
Hoyt Lougee	Foliage Software Systems
Howard Lowe	GE Aviation
Hauke Luethje	NewTec GmbH
Jonathan Lynch	Honeywell
Françoise Magliozzi	Atos Origin
Veronique Magnier	EASA
Kristine Maine	Aerospace Corporation
Didier Malescot	DSNA/DTI
Varun Malik	Hamilton Sundstrand
Patrick Mana	EUROCONTROL
Joseph Mangan	Coanda Aerospace Software
Ghilaine Martinez	DGA Techniques Aéronautiques
Steven C. Martz	Garmin International
Peter Matthews	Independent Consultant
Frank McCormick	Certification Services Inc
Scott McCoy	Harris Corporation
Thomas McHugh	FAA
William McMinn	Lockheed Martin
Josh McNeil	US Army AMCOM SED
Kevin Meier	Cessna Aircraft Company
Amanda Melles	Bombardier
Marc Meltzer	Belcan Engineering
Steven Miller	Rockwell Collins
Gregory Millican	Honeywell
John Minihan	Resource Group
Martin Momberg	Cassidian Air Systems
Pippa Moore	UK CAA
Emilio Mora-Castro	EASA
Endrich Moritz	Technical University
Robert Morris	CDL Systems Ltd.
Allan Terry Morris	NASA
Steve Morton	TBV Associates
Nadir Mostefat	Mannarino Systems
Fred B. Moyer	Rockwell Collins
Robert D. Mumme	Embedded Plus Engineering
Arun Murthi	AERO&SPACE USA
Armen Nahapetian	Teledyne Controls
Gerry Ngu	EASA
Elisabeth Nguyen	Aerospace Corporation
Robert Noel	Mitre Corp
Sven Nordhoff	SQS AG
Paula Obeid	Embedded Plus Engineering
Eric Oberle	Becker Avionics
Brenda Ocker	FAA
Torsten Ostermeier	Bundeswehr
Frederic Painchaud	Defence Research and Development Canada

Name	Organization
Sean Parkinson	Resource Group
Dennis Patrick Penza	AVISTA
Jean-Phillipe Perrot	Turbomeca
Robin Perry	GE Aviation
David Petesch	Hamilton Sundstrand
John Philbin	Northrop Grumman Integrated Systems
Christophe Piala	Thales Avionics
Cyril Picard	EADS APSYS
Francine Pierre	Thales Avionics
Patrick Pierre	Thales Avionics
Gerald Pilj	FAA
Benoit Pinta	Intertechnique - Zodiac
Mo Piper	Boeing Company
Andreas Pistek	ITK Engineering AG
Laurent Plateaux	DGA
Laurent Pomies	Independent Consultant
Jennifer Popovich	Jeppesen Inc.
Clifford Porter	Aircell LLC
Frédéric Pothon	ACG Solutions
Bill Potter	The MathWorks Inc
Sunil Prasad	HCL Technologies, Chennai, India
Paul J. Prisaznuk	ARINC-AEEC
Naim Rahmani	Transport Canada
Angela Rapaccini	ENAC
Lucas Redding	Silver-Atena
David Redman	Aerospace Vehicle Systems Institute (AVSI)
Tammy Reeve	Patmos Engineering Services, Inc.
Guy Renault	SAGEM DS Safran Group
Leanna Rierson	Digital Safety Consulting
George Romanski	Verocel
Cyrille Rosay	EASA
Edward Rosenbloom	Kollsman, Inc
Tom Roth	Airborne Software Certification Consulting
Jamel Rouahi	CEAT
Marielle Roux	Rockwell Collins France
Rudy Ruana	RTCA, Inc.
Benedito Massayuki Sakugawa	ANAC Brazil
Almudena Sanchez	GMV SA
Vdot Santhanam	Boeing Company
Laurence Scales	Thales
Deidre Schilling	Hamilton Sundstrand
Ernst Schmidt	Bundeswehr
Peter Schmitt	Universität Karlsruhe
Dr. Achim Schoenhoff	EADS Military Aircraft
Martin Schwarz	TT Technologies
Gabriel Scolan	SAGEM DS Safran Group
Christel Seguin	ONERA
Beatrice Sereno	Teuchos SAFRAN
Phillip L. Shaffer	GE Aviation
Jagdish Shah	Parker
Vadim Shapiro	TetraTech/AMT

Name	Organization
Jean François Sicard	DGA Techniques Aéronautiques
Marten Sjoestedt	Saab AB
Peter Skaves	FAA
Greg Slater	Rockwell Collins
Claudine Sokoloff	Atos Origin
Marge Sonnek	Honeywell
Guillaume Soudain	EASA
Roger Souter	FAA
Robin L. Sova	FAA
Richard Spencer	FAA
Thomas Sperling	The Mathworks
William StClair	LDRA
Roland Stalford	Galileo Industries Spa
Jerry Stamatopoulous	Aircell LLC
Tom Starnes	Cessna Aircraft Company
Jim Stewart	NATS
Tim Stockton	Certon
Victor Strachan	Northrop-Grumman
John Strasburger	FAA
Margarita Strelnikova	Sukhoi Civil Aircraft Company (SCAC)
Ronald Stroup	FAA
Will Struck	FAA
Wladimir Terzic	SAGEM DS Safran Group
Wolfgang Theurer	C-S SI
Joel Thornton	Tier5 Inc
Mikael Thorvaldsson	KnowIT Technowledge
Bozena Brygida Thrower	Hamilton Sundstrand
Christophe Travers	Dassault Aviation
Fay Trowbridge	Honeywell
Nick Tudor	Tudor Associates
Silpa Uppalapati	FAA
Marie-Line Valentin	Airbus
Jozef Van Baal	Civil Aviation Authorities Netherlands
John Van Leeuwen	Sikorsky Aircraft
Aulis Viik	NAV Canada
Bertrand Voisin	Dassault Aviation
Katherine Volk	L-3 Communications
Dennis Wallace	FAA
Andy Wallington	Bell Helicopter
Yunming Wang	Esterel Technologies
Don Ward	AVSI
Steve Ward	Rockwell Collins
Patricia Warner	Software Engineering
Michael Warren	Rockwell Collins
Rob Weaver	NATS
Yu Wei	CAA China
Terri Weinstein	Parker Hannifin
Marcus Weiskirchner	EADS Military Aircraft
Daniel Weisz	Sandel Avionics, Inc.
Rich Wendlandt	Quantum3D
Michael Whalen	Rockwell Collins

Name	Organization
Paul Whiston	High Integrity Solutions Ltd
Todd R. White	L-3 Communications/Qualtech
Virginie Wiels	ONERA
ElRoy Wiens	Cessna Aircraft Company
Terrance Williamson	Jeppesen Inc.
Graham Wisdom	BAE Systems
Patricia Wojnarowski	Boeing Commercial Airplanes
Joerg Wolfrum	Diehl Aerospace
Kurt Woodham	NASA
Cai Yong	CAAC (Civil Aviation Administration of China)
Edward Yoon	Curtiss-Wright Controls, Inc
Robert Young	Rolls-Royce
William Yu	CAAC China
Erhan Yuceer	Savunma Teknolojileri Muhendislik ve Ticaret
Uli Zanker	Liebherr

APPENDIX MB.B – FREQUENTLY ASKED QUESTIONS AND DISCUSSION PAPERS

This appendix contains Frequently Asked Questions (FAQs) (sections MB.B.1 through MB.B.16) and Discussion Papers (DPs) (MB.B.17 through MB.B.18) specific to Model-Based Development and Verification. Applicability to DO-278A is included in the FAQs and DPs for completeness and for consistency with FAQs and DPs in other documents (for example, DO-248C).

MB.B.1 FAQ #1: What is the Data Control Category of a model?

Reference: This document: Section MB.4.2
DO-178C/DO-278A: Section 7

Keywords: control category; Design Model; Specification Model

Answer:

The Data Control Category of a model is the same as the life cycle data the model represents. For example, a Design Model representing software architecture would be assigned to CC1 for software levels A, B, and C; for software level D, a Design Model would be assigned to CC2.

MB.B.2 FAQ #2: Which verification objectives does model simulation, by itself support?

Reference: This document: Sections MB.6.7.2 and MB.6.8.1

Keywords: model simulation

Answer:

As indicated in MB.6.8.1, model simulation can be used to fully satisfy some of the objectives of Annex MB.A Table MB.A-3 and Table MB.A-4 (Annex MB.C Table MB.C-3 and Table MB.C-4 for DO-278A users).

For those objectives, this activity may be used to replace reviews and/or analysis of the Design Model.

Note: This FAQ references objectives through the Annex Tables. Although the Annex Tables contain a summary of the objectives (for ease of reference), the entire objective within the body of the text should be considered.

MB.B.3 FAQ #3: What verification objectives can be met by combining model simulation with other tools or methods?

Reference: This document: Sections MB.6.7.2, MB.6.8, and MB.6.8.2

Keywords: model simulation; planning; tools; tool qualification

Answer:

As explained in section MB.6.8.2, model simulation can be used in combination with other tools and methods (for example, model coverage analysis and resolution, a qualified autocode generator, and an "accepted" or "acceptable" compiler), to satisfy some

objectives of Annex MB.A Tables MB.A-6 and Table MB.A-7 (Annex MB.C Table MB.C-6 and Table MB.C-7 for DO-278A users).

Rationale for how the tool chain and process supports the objectives should be included in the software plans.

Note: This FAQ references objectives through the Annex Tables. Although the Annex Tables contain a summary of the objectives (for ease of reference), the entire objective within the body of the text should be considered.

MB.B.4 FAQ #4: If a model is used to represent requirements and an autocode generator is not used, does the supplement apply?

Reference: This document: Sections MB.4, MB.5, and MB.6

Keywords: Software Model Standards; Source Code

Answer:

If a model is used to represent requirements, then the guidance of this supplement applies. The method of creating Source Code, either manually or automatically, does not change the applicability of DO-178C/DO-278A or this supplement.

When a model is used to represent requirements, then the objectives, activities (such as model coverage analysis), and software data items (such as Software Model Standards) are important, regardless of how the Source Code is generated. The planning process should address each activity and objective as it relates to model usage within the software life cycle.

MB.B.5 FAQ #5: How should the applicant develop and verify manually written Source Code, that is invoked by the code generated from the Design Model?

Reference: DO-178C/DO-278A: Section 6.4

Keywords: coverage analysis; development; planning; testing

Answer:

Although the supplement does not have guidance on this topic, the following information should be considered when using DO-178C/DO-278A.

Manually written Source Code may be invoked from the Source Code generated from a Design Model in a variety of ways. Such manually written Source Code should be developed and verified as any other manually written Source Code. Activities of DO-178C/DO-278A include:

- Development and verification of the low-level requirements for this Source Code.
- Verification of the software with respect to the objectives of DO-178C/DO-278A Annex A Tables A-5, A-6 and A-7.

Integration testing of this code with code generated from the Design Model should also be performed.

This applies to software provided by a tool vendor, as well as software developed by a tool user. In both cases, verification evidence should be provided as described in DO-178C/DO-278A.

The software plans should address this manually generated Source Code, including development, configuration management, and verification including testing.

Note: This FAQ references objectives through the Annex Tables. Although the Annex Tables contain a summary of the objectives (for ease of reference), the entire objective within the body of the text should be considered.

MB.B.6 FAQ #6: When Source Code is automatically generated from a Design Model, what type of testing should be performed to provide assurance of the correctness of the integration of this Source Code with manually written Source Code?

Reference: DO-178C/DO-278A: Section 6.4

Keywords: coverage analysis; development; planning; testing

Answer:

Source Code automatically generated from a Design Model may be invoked from manually written Source Code in a variety of ways, such as calls from a scheduler or operating system. Likewise, automatically generated Source Code may also invoke manually written Source Code.

Testing of the Executable Object Code resulting from the integration of the automatically generated Source Code with manually written Source Code should be performed as described in DO-178C/DO-278A sections 6.4.3.a and 6.4.3.b. This would ensure the following:

- Both types of code interact correctly.
- Executable Object Code resulting from the integration complies with the software architecture.

MB.B.7 FAQ #7: When using models for verification, should expected results be determined prior to test execution?

Reference: DO-178C/DO-278A: Section 6.4

Keywords: expected results; test cases

Answer:

When using models for verification, expected results should be determined prior to test execution.

An expected result need not be a specific value; it could be provided in an alternative form, like an equation or a model. As an example, the expected result for calibrated airspeed may be defined in a model as the equation:

Cal_AirSpd= 661.4788(5(((Pitot_Pressure)**(2/7)+1)-1))**0.5

Providing an alternative form for an expected result would be appropriate when the test stimulus cannot be predetermined due to the required results accuracy prior to the test execution. This can be understood for the above airspeed equation when practical limitations of the precision of setting a simulated pitot pressure makes it impossible to create a highly precise "expected result" prior to running the actual test procedure.

By allowing the expected result to be evaluated based on the expected result equation, a highly precise "expected versus actual" result can be obtained by the testing activity. This technique can allow for execution of highly accurate test procedures without having to use excessively wide tolerances that would otherwise be required to compensate for an imprecise input set point.

The alternative form, whether in equation or model form, should be traceable and compliant with the requirements from which the model was developed. This traceability and compliance will be accomplished as part of the existing test review activities.

Note: Care should be taken to ensure that the calculations of the expected results reflect the software requirements, not the code under test. For example, this can be achieved by using different library functions or greater resolution of data types used to calculate expected results.

MB.B.8 FAQ #8: Is a model subject to tool qualification?

Reference: This document: Sections MB.4.4.3, MB.4.4.4, and MB.12.2
DO-178C/DO-278A: Section 4.4.1

Keywords: tool qualification

Answer:

A model is an abstract representation of certain aspects of a system and is not a tool; therefore, the model itself is not subject to tool qualification.

However, there are generally tools associated with models. See section MB.12.2 to determine if tool qualification is needed for any tools associated with the models.

MB.B.9 FAQ #9: What is an example of a model that is considered part of the test environment and what activities are applicable to that model?

Reference: This document: Sections MB.11.3.d, MB.11.15.c, and MB.12.2
DO-178C/DO-278A: Sections 4.4.3 and 6.4.1

Keywords: test environment

Answer:

Although this supplement does not have guidance on this topic, the following information should be considered when using DO-178C/DO-278A.

Examples of models that are considered part of the test environment may include models that are used to represent the behavior of the aircraft or engine.

A model that represents the dynamic behavior of the engine (sometimes referred to as a plant model) is a specific example of a model used in a test environment. This model, as

used as part of the test environment, should be properly assessed to see that it meets the fit-for-purpose testing intent; that is, working within the accepted limitations of the model with respect to real-world behavior.

DO-178C/DO-278A section 4.4.3 requires that the planning cover all aspects of the test environment. Differences and possible limitations of the test environment's model should be properly assessed during planning to assure that the test environment models properly represent real-world environment. Configuration management of models used for verification is also important.

The Software Verification Plan should address the models that are part of the test environment (see MB.11.3.d). The Software Life Cycle Environment Configuration Index should identify the models that are part of the test environment (see MB.11.15.c).

In summary, the models used in a test environment should be planned, identified, assessed, and controlled just like any other aspect of the test environment.

MB.B.10 FAQ #10: Can a single Software Model Standard be applied to both Specification Models and Design Models?

Reference: This document: Sections MB.4.2, MB.11.23.b, and MB.11.23.c

Keywords: Software Model Standard; software planning

Answer:

No. Specific aspects within the Software Model Standard (see MB.11.23) require distinctions based on the different software life cycle data models represent. The planning process in MB.4.2 indicates that usage of these models and their associated standards should include a rationale that justifies how and why the selected modeling technique supports the specific artifact (for example, a Specification Model or a Design Model). The two Software Model Standards may be packaged in a single document.

MB.B.11 FAQ #11: May the applicant use the model coverage analysis activity to achieve the structural coverage analysis objectives?

Reference: This document: Sections MB.6.7, MB.6.8.1, and MB.6.8.2
DO-178C/DO-278A: Section 6.4.4.2

Keywords: model coverage analysis; structural coverage analysis

Answer:

This FAQ assumes that model coverage analysis has been performed as defined in MB.6.7.

As noted in MB.6.7, "Model coverage analysis is different than structural coverage analysis and therefore model coverage analysis does not eliminate the need to achieve the objectives of structural coverage analysis per DO-178C section 6.4.4.2." However, model coverage analysis may be used as a means for achieving structural code coverage analysis objectives under appropriate conditions. In that case, the applicant should include the plan for this approach in the Software Verification Plan and capture the actual demonstration/verification as part of the Software Verification Results. These conditions should be evaluated on a case-by-case basis and agreed upon by the certification

authorities, in order to take into account specific aspects of the modeling notation, model coverage criteria, and code generation characteristics.

These conditions should include at least the following:

- Model coverage analysis criteria hold the same properties as the applicable structural code coverage analysis criteria hold for the level of the software being developed, for example, MC/DC coverage for the level A.
- Qualification of the code generation tool chain with respect to objectives for which certification credit is sought (in particular Annex MB.A Table MB.A-7 (Annex MB.C Table MB.C-7 for DO-278A users)) should preserve the applicable coverage criteria.
- Any libraries used by code generated from the Design Model are verified according to DO-178C/DO-278A section 6, including structural code coverage activities in accordance with the required software level.

MB.B.12 FAQ #12: What are the independence issues when a Design Model is used for both code generation and test generation?

Reference: This document: Sections MB.4 and MB.12.2
DO-178C/DO-278A: Section 6.4

Keywords: code generation; Design Model; independence; test generation

Answer:

If a single model representing requirements data is used as input to both the code generation and test generation processes, those two downstream processes should be independent to satisfy objectives 3 and 4 of Annex MB.A Table MB.A-6 (Annex MB.C Table MB.C-6 for DO-278A users).

There are multiple approaches when Source Code and test cases used to verify the Executable Object Code are automatically generated. Some approaches may be used on one unique tool set or rely upon several tools. The certification credit that can be sought from automatic generation (code or test) may also differ between approaches, depending on tool qualification strategy.

Whatever the scenario, if both Source Code and test cases are automatically generated from a Design Model, the independence objectives of Annex MB.A Table MB.A-6 (Annex MB.C Table MB.C-6 for DO-278A users) remain applicable.

Note: This FAQ references objectives through the Annex Tables. Although the Annex Tables contain a summary of the objectives (for ease of reference), the entire objective within the body of the text should be considered.

MB.B.13 **FAQ #13: How does the supplement apply if a Design Model that is used in the software development process is part of the system-level life cycle data, as in example 5 of Table MB.1-1 (that is, the Design Model also contains system requirements allocated to software)?**

Reference: This document: Sections MB.1.6.1, MB.5, MB.6, and MB.B.17 (DP#1); and Table MB.1-1
DO-178C: Section 2.6

DO-278A: Section 2.7

Keywords: Design Model; system level; system requirements

Answer:

If a Design Model used in the software development process is part of the system life cycle data, this supplement applies. As a consequence, the guidance related to low-level requirements should be applied to the requirements contained in the Design Model and the guidance related to high-level requirements should be applied to the requirements from which the Design Model is developed. Therefore, in terms of objectives and activities required by this supplement, MB Example 5 in Table MB.1-1 is equivalent to MB Example 1 in Table MB.1-1.

In Table MB.1-1, the difference between models in Examples 1 and 5 is that some activities may be performed by using equivalent guidance as part of the system development process to fulfill the objectives of Annex MB.A Table MB.A-3 (Annex MB.C Table MB.C-3 for DO-278A users). In such a case, the demonstration of equivalence between the alternate guidance and the guidance of DO-178C/DO-278A and this supplement should be substantiated; the associated process and activities included in the software plans; and the guidance of DO-178C section 2.6 (or DO-278A section 2.7) should be followed.

See MB.B.17 DP#1, for further information.

Note: This FAQ references objectives through the Annex Tables. Although the Annex Tables contain a summary of the objectives (for ease of reference), the entire objective within the body of the text should be considered.

MB.B.14 **FAQ #14: How does the supplement apply if the requirements from which the Design Model was developed are part of the system life cycle data, as in example 4 and 5 of Table MB.1-1 (that is, high-level requirements per DO-178C/DO-278A and this supplement are at the system level in the form of system requirements allocated to software)?**

Reference: This document: Sections MB.1.6.1, MB.5, MB.6, and MB.B.17 (DP#1); and Table MB.1-1
DO-178C: Section 2.6
DO-278A: Section 2.7

Keywords: Design Model; system level; system requirements

Answer:

If the requirements from which the Design Model is developed are part of the system life cycle data, this supplement applies to those requirements. As a consequence, the guidance related to low-level requirements should be applied to the requirements contained in the Design Model, and the guidance related to high-level requirements should be applied to the requirements from which the Design Model is developed. Therefore, in terms of objectives and activities required by this supplement, MB Examples 4 and 5 in Table MB.1-1 are equivalent to MB Example 1 in Table MB.1-1.

In Table MB.1-1, the difference between models in Example 1 and Example 4 and 5 is that the requirements from which the Design Model is developed can be developed, controlled, and verified using equivalent guidance as part of the system development process to fulfill the objectives of Annex MB.A Table MB.A-3 (Annex MB.C Table MB.C-3 for DO-278A users). In such a case, the demonstration of equivalence between the alternate guidance and the guidance of DO-178C/DO-278A and this supplement should be substantiated, the associated process and activities included in the software plans, and the guidance of DO-178C section 2.6 (or DO-278A section 2.7) should be followed.

If the requirements from which the Design Model is developed are outputs of the system process, the objectives 1 and 6 of Annex MB.A Table MB.A-3 (Annex MB.C Table MB.C-3 for DO-278A users) are implicitly met for those requirements.

See MB.B.17 DP#1, for further information.

Note: This FAQ references objectives through the Annex Tables. Although the Annex Tables contain a summary of the objectives (for ease of reference), the entire objective within the body of the text should be considered.

MB.B.15 FAQ #15: Do data files associated with models need to be treated as parameter data items?

Reference: DO-178C/DO-278A: Sections 6.6 and 11.22

Keywords: parameter data item; Parameter Data Items File

Answer:

If a data file is associated with a model, and the parameters within that file are used to produce the Executable Object Code from the model, then that file does not need to be treated as a parameter data item. In this case, the Executable Object Code and parameters will be verified together.

If parameters within data files associated with a model are used to produce a Parameter Data Item File, then that file does need to be treated as a parameter data item. Per the definition, a Parameter Data Item File is separate from the Executable Object Code. In this case, the Parameter Data Item File and the Executable Object Code may be verified together or separately in accordance with the guidance in DO-178C/DO-278A section 6.6.

MB.B.16 FAQ #16: Can simulation support the assessment of test coverage of the low-level requirements contained in a Design Model?

Reference: This Document: Sections MB.6.7 and MB.6.8.2
DO-178C/DO-278A: Section 6.4

Keywords: model coverage analysis; simulation

Answer:

There exists a specific instance where simulation of low-level requirements contained in a Design Model could support the assessment of test coverage. The constraints for this specific case are as follows:

- High-level requirements-based tests are developed.
- These tests are run on the Executable Object Code to verify compliance of the Executable Object Code to the high-level requirements.
- These same tests are used for the simulation of the Design Model to verify compliance of the Design Model to the high-level requirements.

In this case, simulation in combination with model coverage analysis, can support the assessment of test coverage of the low-level requirements contained in the Design Model.

When this approach is used, the high-level requirements-based tests are run on the Executable Object Code; therefore, MB.6.8.2 does not apply.

MB.B.17 DP #1: Examples of model-based development and the relationship between a Design Model or a Specification Model and DO178C high-level requirements, low-level requirements, and software architecture.

Reference: This document: Sections MB.1.6.1 and MB.2.1.1

Keywords: architecture; Design Model; high-level requirements; low-level requirements; architecture; model; model-based development; Specification Model

MB.B.17.1 Background:

Relating models to traditional software life cycle processes and data has been subjective and therefore a wide range of differing interpretations have been made. This DP will provide examples to correlate model usage with DO-178C/DO-278A concepts.

MB.B.17.2 Discussion:

The figures below help relate DO-178C/DO-278A concepts of high-level requirement, low-level requirement, and software architecture. Examples of some of the possible uses of models within the development process are shown in this DP.

Acronyms used in this DP:

- HLR High-Level Requirements
- LLR Low-Level Requirements
- SRATS System Requirements allocated to Software
- SW Architecture Software Architecture

Any requirement described as "textual" in the context of this DP is called "textual" to make clear that it is not a requirement represented by a model, and it could be given in any other possible form.

Note: This DP references objectives through the Annex Tables. Although the Annex Tables contain a summary of the objectives (for ease of reference), the entire objective within the body of the text should be considered.

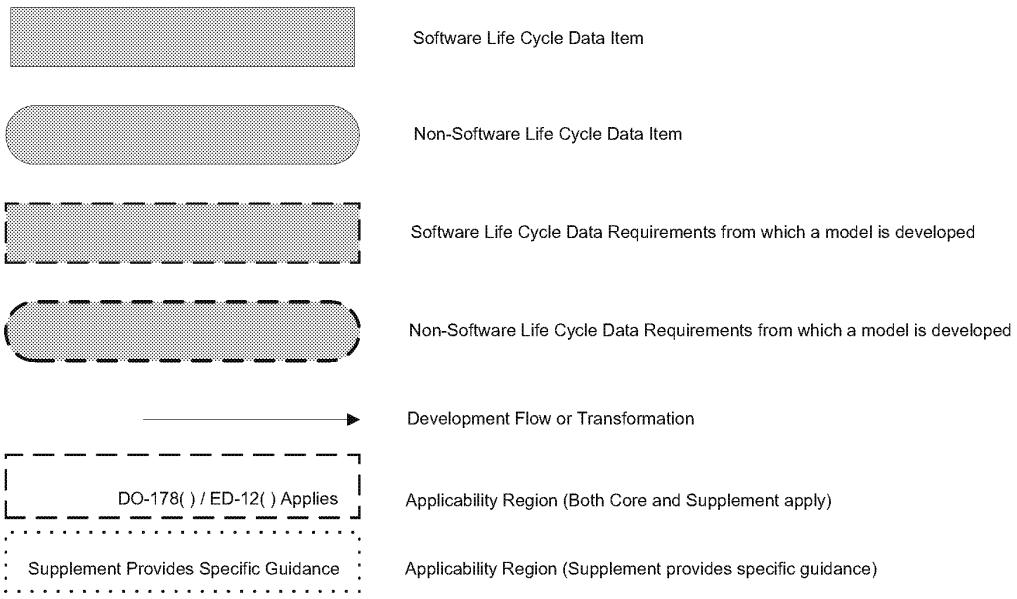

FIGURE MB.DP1-1
KEY TO READING DP1 DIAGRAMS

Note: The bounding white region shows the area of applicability of DO-178C/DO-278A and this supplement. This supplement applicability in these regions generally consists of mapping DO-178C/DO-278A objectives and activities into a Model-Based Development and Verification (MB) context using this technology's terminology and conventions. Conversely, the grey bounded region identifies software life cycle data for which specific guidance is provided within this supplement.

MB.B.17.3 Example A

This example is the same as Example 1 in Table MB.1-1 of this supplement; one (or several) Design Model(s) may be used to represent the LLRs and the SW architecture.

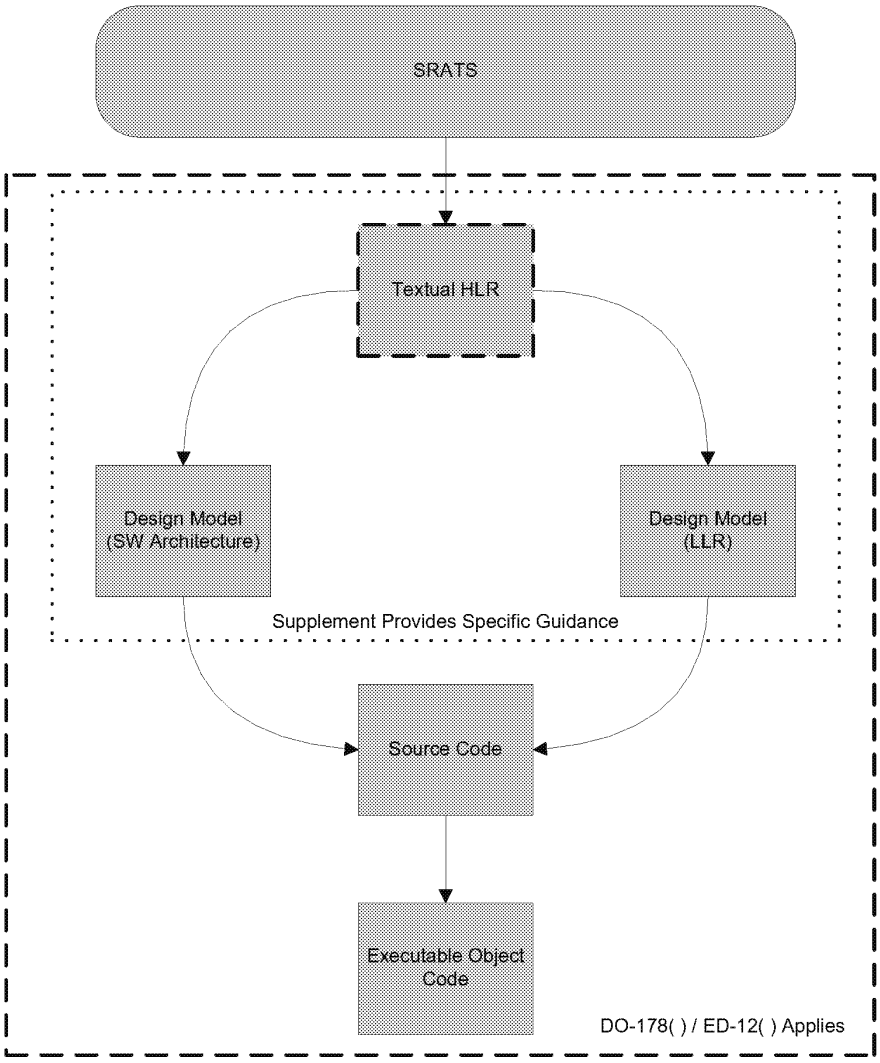

Figure MB.DP1-2
Example A: Separate Models Used To Express LLR and SW Architecture

MB.B.17.4 Example B

This example is similar to Example 4 in Table MB.1-1 of this supplement; it is possible to have just one model that represents the software LLR and software architecture, with the requirements from which the Design Model is developed being created by the system processes.

Note: In that case, objectives for HLR apply to the requirements from which the Design Model is developed. Those system requirements are part of the SRATS and this supplement only applies to this part of the SRATS.

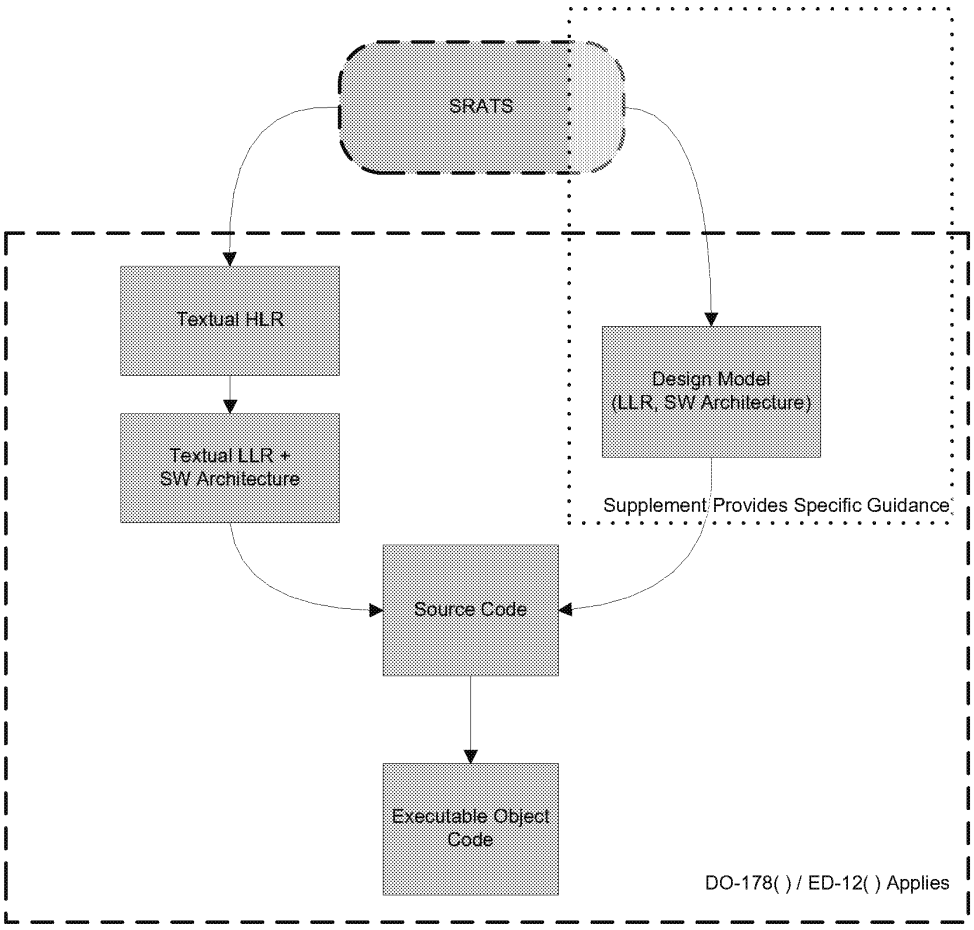

Figure MB.DP1-3
Example B: One Model Used To Express LLR/SW Architecture with HLR Provided By SRATS

Note: In this example, the supplement provides guidance for artifacts developed in the system processes. The portion of the SRATS from which the Design Model was developed is considered to be the software HLRs, and therefore will need to comply with objectives of Annex MB.A Table MB.A-3 (Annex MB.C Table

MB.C-3 for DO-278A users). Since the SRATS are also the HLRs, compliance and traceability between SRATS and HLRs are met by construction, therefore Annex MB.A Table MB.A-3 (Annex MB.C Table MB.C-3 for DO-278A users) objectives 1 and 6 are inherently satisfied.

MB.B.17.5 Example C

This example is similar to Example 2 in Table MB.1-1 of this supplement. Additionally, it is possible to have one model that represents the HLRs and a different model that represents the LLRs and software architecture.

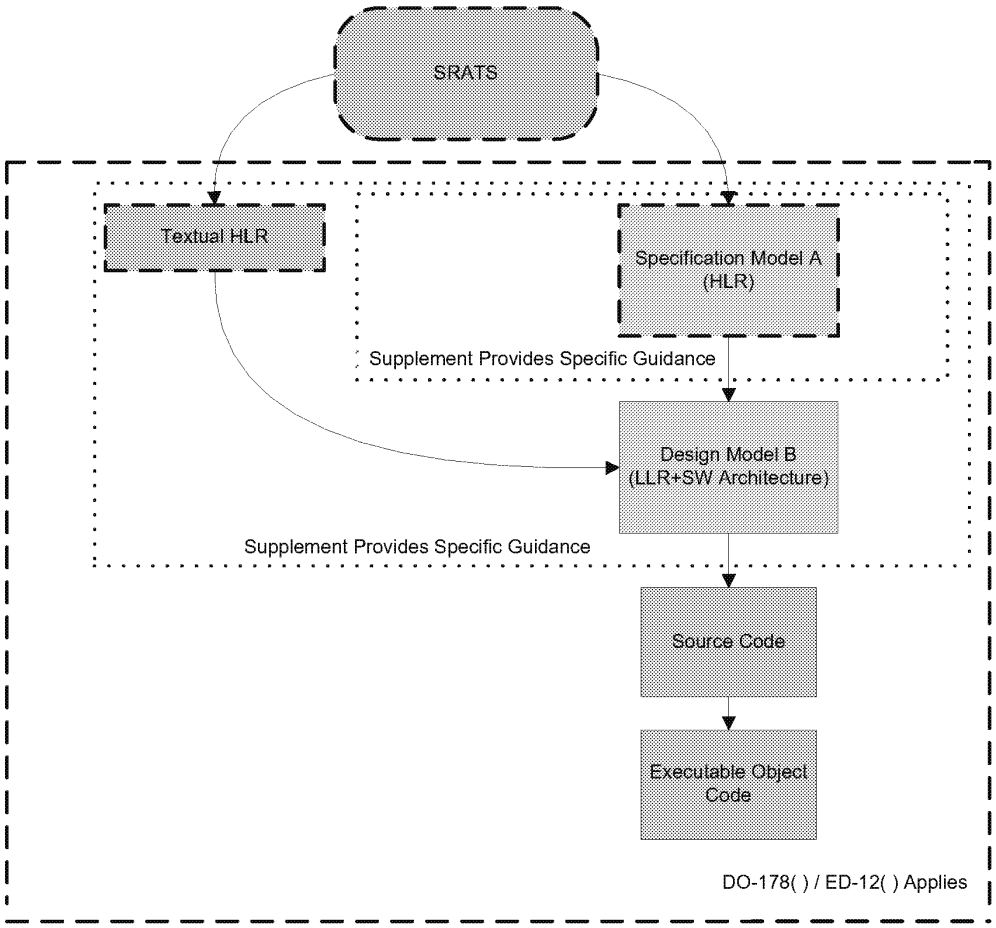

Figure MB.DP1-4
Example C: Separate Models Used To Express HLR and LLR/SW Architecture

MB.B.17.6 Example D

This example is similar to Example 3 in Table MB.1-1 of this supplement. It is possible to have one model that represents a part of the HLRs and a "classical" development below this model.

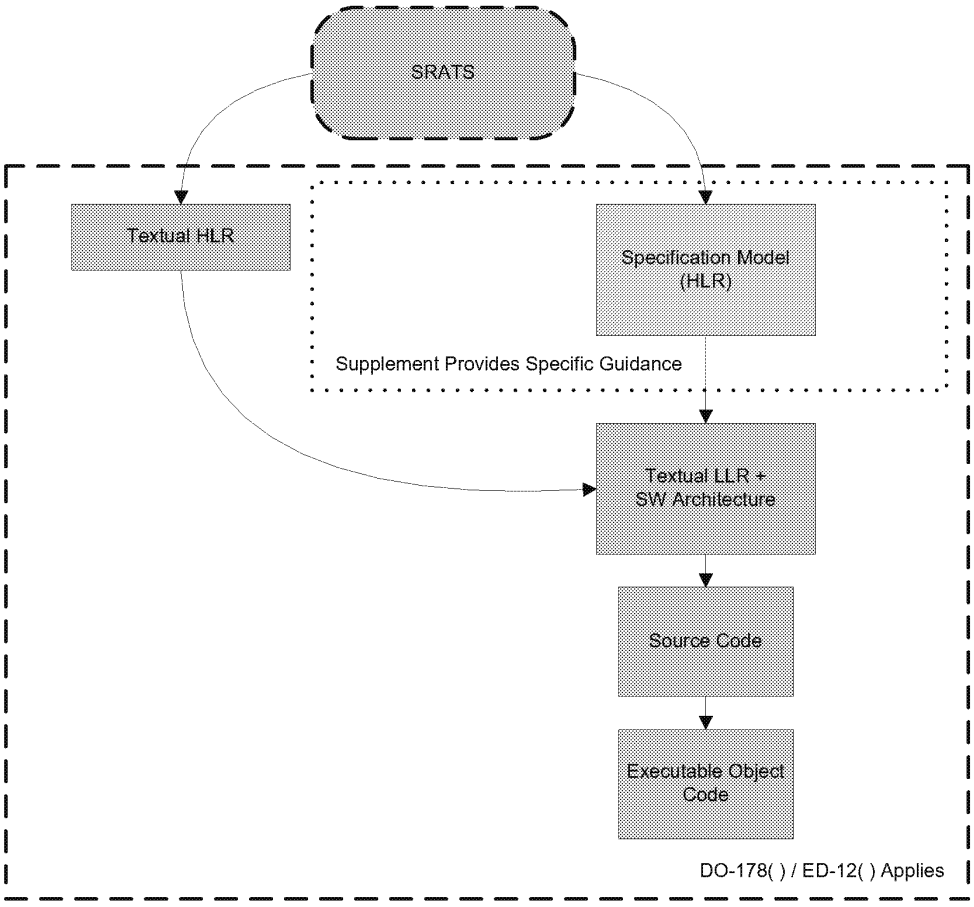

Figure MB.DP1-5
Example D: One Model Used To Express HLR and Only HLR

MB.B.17.7 Example E

This example shows the case where System Requirements allocated to Software are provided as a Design Model, as described in section MB.2.1.1. This is similar to Example 5 in Table MB.1-1 of this supplement, where the requirements from which the Design Model is developed are created by the system processes.

Note: In that case, objectives for HLR apply to the requirements from which the Design Model is developed. Those requirements are part of the SRATS and this supplement only applies to the part of the SRATS from which the Design Model was developed.

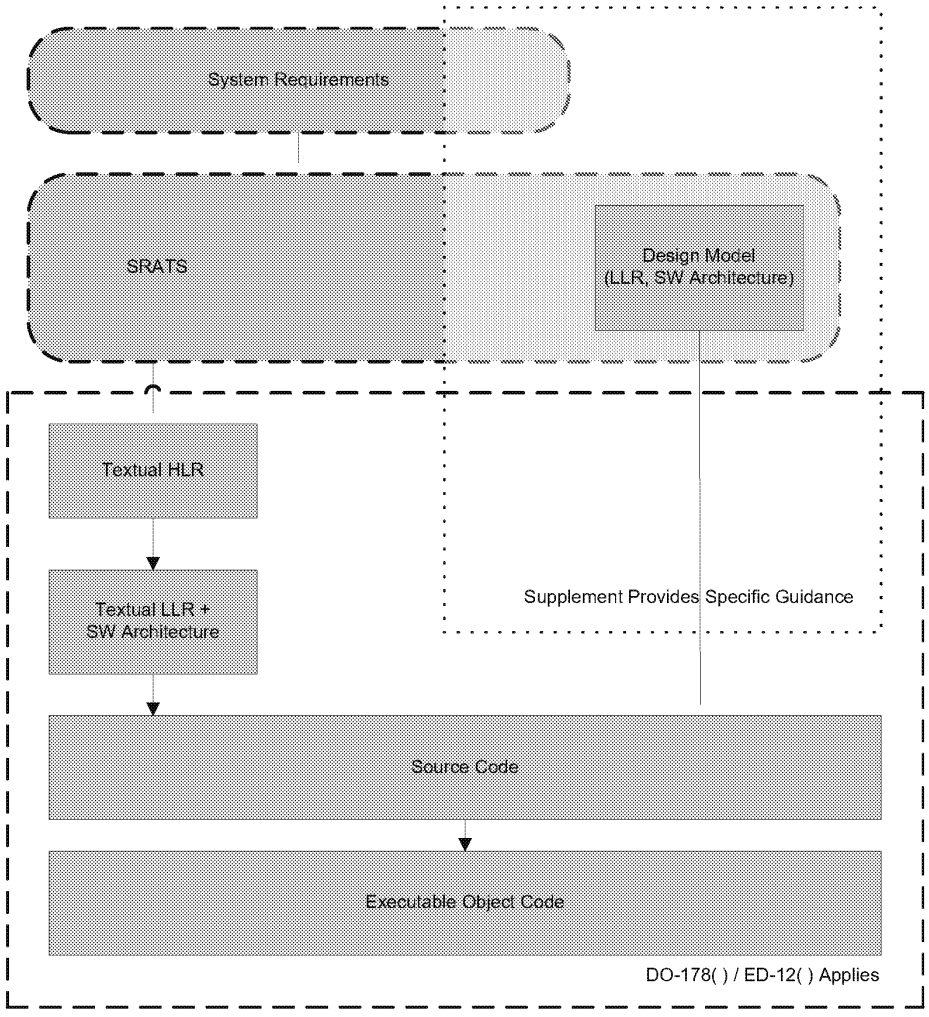

Figure MB.DP1-6
Example E: One Model Provided By SRATS Used To Express LLR/SW Architecture

Note: In this example, the supplement provides guidance for the artifacts developed in the system processes. Although the Design Model was developed in the system processes and is part of the SRATS, the subset of the SRATS expressed as a Design Model will need to comply with the objectives of Annex MB.A Table

MB.A-4 (Annex MB.C Table MB.C-4 for DO-278A users). As shown in the example, this subset of SRATS is also considered the software LLRs. The subset of system requirements above the SRATS from which the Design Model was developed will need to comply with the objectives of Annex MB.A Table MB.A-3 (Annex MB.C Table MB.C-3 for DO-278A users). Since the system requirements are also the HLRs, compliance and traceability between system requirements and HLRs are met by construction, therefore Annex MB.A Table MB.A-3 objectives 1 and 6 (Annex MB.C Table MB.C-3 objectives 1 and 6 for DO-278A users) are inherently satisfied.

MB.B.18 DP #2: Information on the usage of libraries in a Model-Based Development and Verification processes.

Reference: This document: Sections MB.5.0 and MB.6.0

Keywords: library; model element library; symbol

MB.B.18.1 Background

The use of software libraries in general has been an area with varying interpretations. The usage of libraries in conjunction with models has complicated interpretations even more. This DP will provide information to help clarify the usage of libraries in model-based development and verification processes.

MB.B.18.2 Discussion

Model-Based Development and Verification processes frequently involve use of libraries, called model element libraries throughout this supplement. To ease the readability, within this DP, the single word "library" is often used in place of "model element library". The use of libraries as such is not new and already existed in traditional processes. What changes with the use of models is the following:

- In many cases, libraries are at the heart of Model-Based Development and Verification processes, utilized in modeling, verification, coding, and testing.
- These libraries are frequently developed by third parties, such as tool vendors, department other than the one using the library, etc.
- Some libraries are directly developed in a programming language and others are developed in a modeling language.

This paper applies to library elements that are used for developing airborne software.

MB.B.18.3 Library Life cycle Data and Processes

Independently of who develops the library and who uses it, the following principles should apply:

- The processes and activities used for development and verification of libraries should be compliant with DO-178C/DO-278A objectives for the level of the application for which they are used.

- All library life cycle data required by DO-178C/DO-278A for the level of the application should be available to the library user and certification authority.
- The library developer should provide documentation of the library (see details in next sections of this DP).
- The library user should document his operational requirements for the library.

Note: If the library is developed by the applicant during the software life cycle, operational requirements may be contained in the library requirements.

MB.B.18.4 Library Requirements

For any library, there should be a set of requirements from which the library is developed.

Since library elements may be used in a variety of contexts, special care should be taken to explicitly state the conditions under which the library elements may be used. Derived requirements should be precisely documented and made available to the library user. Library requirements include, but are not limited to:

a. Interface conventions (such as data types, data passing modes).

b. Constraints on input data. An example of a constraint is a limit on the value of an input parameter.

c. Memory management.

d. Scheduling conditions.

e. Identification of system resources.

f. Any known limitations.

MB.B.18.5 Library Configuration Management Process

The control category of library life cycle data during library development and verification should be that of the control category of the most critical software level of the application using the library.

The control category of library life cycle data during library use should be that of the application using the library.

MB.B.18.6 Library Standards

The two different types of library standards that need to be considered are:

a. Standards for Library Development and Verification

The modeling /design standards of the library development should be provided to the library user. This includes:

- Internal design/coding rules.

- Precise rules for interfacing libraries with the model-based environment and code (for example, how data is passed and how memory is managed).

b. Standards for Library Use

The modeling/design standards used in the project software development processes that use the libraries should provide precise rules for interfacing libraries with the model-based environment and code. This may be achieved by references to the library documentation, possibly complemented by application specific rules.

The library user should verify compatibility (not necessarily identify) of the design/coding standards of the library development with those of the application development.

MB.B.18.7 Coverage of DO-178C/DO-278A Objectives for Libraries

The library user should provide evidence that the activities performed by the library developer and library user combine to ensure that all objectives of the libraries and application are satisfied.

Note: This paper does not impose any specific sharing of tasks among the library developer and the library user.

Verification objectives (see MB.6) include but are not limited to:

- Verification of library against its requirements (including reviews and testing).
- Review library usage by the application (design and code reviews).
- Integration testing between the application and the library.

Library elements fall into one of the following categories:

- Model-based library element: Such an element is developed using a modeling language. In this case, this supplement is applicable.
- Non model-based library element: Such an element is not developed using a modeling language. In this case, DO-178C/DO-278A is applicable.

MB.B.18.8 Symbology

"Symbology" means the graphical appearance of modeling elements. Some modeling environments allow custom symbology to be defined.

In order to avoid risk of misinterpretation, symbols should fulfill the following requirements:

- They should be uniquely identified.
- They should not be misleading.
- They should be documented (requirements, layout).

There should be standards for symbology, which should be part of Software Model Standards.

There should be verification activities for symbols, in order to verify that the above mentioned requirements are satisfied.

MB.B.18.9 Partial Use of Libraries

There may be cases where the following are applicable:

- Some library elements are not used at all in an application (for example, the application uses sine but not cosine from a trigonometric library).
- For some library elements, some functionalities are not used (for example, the "reset" function of a filter).

If libraries are only partially used, the applicant should perform an analysis to detect whether the partially used elements may lead to dead code or deactivated code. Refer to handling of dead code or deactivated code in DO-178C/DO-278A.

MB.B.18.10 Reuse of Libraries

Refer to previously developed software in DO-178C/DO-278A.

This Page Intentionally Left Blank